Inverse eigenvalue problems for extended Hessenberg and extended tridiagonal matrices

Thomas Mach, Raf Vandebril, and Marc Van Barel

Thomas Mach
Department of Computer Science
KU Leuven, Belgium
Thomas.Mach@cs.kuleuven.be

Raf Vandebril
Department of Computer Science
KU Leuven, Belgium
Raf.Vandebril@cs.kuleuven.be

Marc Van Barel
Department of Computer Science
KU Leuven, Belgium
Marc.VanBarel@cs.kuleuven.be

Abstract

In inverse eigenvalue problems one tries to reconstruct a matrix, satisfying some constraints, given some spectral information. Here, two inverse eigenvalue problems are solved.

First, given the eigenvalues and the first components of the associated eigenvectors (called the weight vector) an extended Hessenberg matrix with prescribed poles is computed possessing these eigenvalues and satisfying the eigenvector constraints. The extended Hessenberg matrix is retrieved by executing particularly designed unitary similarity transformations on the diagonal matrix containing the eigenvalues. This inverse problem closely links to orthogonal rational functions: the extended Hessenberg matrix contains the recurrence coefficients given the nodes (eigenvalues), poles (poles of the extended Hessenberg matrix), and a weight vector (first eigenvector components) determining the discrete inner product. Moreover, it is also sort of the inverse of the (rational) Arnoldi algorithm: instead of using the (rational) Arnoldi method to compute a Krylov basis to approximate the spectrum, we will reconstruct the orthogonal Krylov basis given the spectral info.

In the second inverse eigenvalue problem, we do the same, but refrain from unitarity. As a result we execute possibly non-unitary similarity transformations on the diagonal matrix of eigenvalues to retrieve a (non)-symmetric *extended tridiagonal* matrix. The algorithm will be less stable, but it will be faster, as the extended tridiagonal matrix admits a low cost factorization of $\mathcal{O}(n)$ (n equals the number of eigenvalues), whereas the extended Hessenberg matrix does not. Again there is a close link with orthogonal function theory, the extended tridiagonal matrix captures the recurrence coefficients of bi-orthogonal rational functions. Moreover, it is again sort of inverse of the nonsymmetric Lanczos algorithm: given spectral properties, we reconstruct the two basis Krylov matrices linked to the nonsymmetric Lanczos algorithm

Article information

- Mach, Thomas; Vandebril, Raf; Van Barel, Marc. Inverse eigenvalue problems for extended Hessenberg and extended tridiagonal matrices, Journal of Computational and Applied Mathematics, volume 272, pages 377-398, 2014.
- The content of this article is identical to the content of the published paper, but without the final typesetting by the publisher.
- Journal's homepage: http://www.journals.elsevier.com/journal-of-computational-and-applied-mathematics/
- Published version: http://dx.doi.org/10.1016/j.cam.2014.03.015
- KU Leuven's repository url: https://lirias.kuleuven.be/handle/123456789/450130

INVERSE EIGENVALUE PROBLEMS FOR EXTENDED HESSENBERG AND EXTENDED TRIDIAGONAL MATRICES *

THOMAS MACH†, MARC VAN BAREL†, AND RAF VANDEBRIL†

Abstract. In inverse eigenvalue problems one tries to reconstruct a matrix, satisfying some constraints, given some spectral information. Here, two inverse eigenvalue problems are solved.

First, given the eigenvalues and the first components of the associated eigenvectors (called the weight vector) an extended Hessenberg matrix with prescribed poles is computed possessing these eigenvalues and satisfying the eigenvector constraints. The extended Hessenberg matrix is retrieved by executing particularly designed unitary similarity transformations on the diagonal matrix containing the eigenvalues. This inverse problem closely links to orthogonal rational functions: the extended Hessenberg matrix contains the recurrence coefficients given the nodes (eigenvalues), poles (poles of the extended Hessenberg matrix), and a weight vector (first eigenvector components) determining the discrete inner product. Moreover, it is also sort of the inverse of the (rational) Arnoldi algorithm: instead of using the (rational) Arnoldi method to compute a Krylov basis to approximate the spectrum, we will reconstruct the orthogonal Krylov basis given the spectral info.

In the second inverse eigenvalue problem, we do the same, but refrain from unitarity. As a result we execute possibly non-unitary similarity transformations on the diagonal matrix of eigenvalues to retrieve a (non)-symmetric *extended tridiagonal* matrix. The algorithm will be less stable, but it will be faster, as the extended tridiagonal matrix admits a low cost factorization of $\mathcal{O}(n)$ (n equals the number of eigenvalues), whereas the extended Hessenberg matrix does not. Again there is a close link with orthogonal function theory, the extended tridiagonal matrix captures the recurrence coefficients of bi-orthogonal rational functions. Moreover, it is again sort of inverse of the nonsymmetric Lanczos algorithm: given spectral properties, we reconstruct the two basis Krylov matrices linked to the nonsymmetric Lanczos algorithm

Key words. rational nonsymmetric Lanczos, rational Arnoldi, inverse eigenvalue problem, data sparse factorizations, extended Hessenberg matrix

AMS subject classifications. 65F18, 65F25, 65F60, 65D32

1. Introduction.

In this manuscript special instances of the following very general inverse eigenvalue problem are solved.

DEFINITION 1.1 (Inverse Eigenvalue Problem, IEP-general). *Given n complex numbers λ_i and corresponding positive real weights w_i, $i = 1, 2, \ldots, n$. Without loss of generality, we will assume that the vector $w = [w_1, w_2, \ldots, w_n]$ has 2-norm equal to 1. Find a matrix M having a certain desired structure such that the eigenvalues of M are λ_i and such that the first component of the corresponding unit eigenvector is w_i.*

This inverse eigenvalue problem computes the recurrence coefficients of orthogonal functions, orthogonal with respect to a discrete inner product with the λ_i as nodes and the $|w_i|^2$ as weights of the inner product. Solving such inverse eigenvalue problems, i.e. computing the recurrences and orthogonal functions (stemming from polynomials, Laurent polynomials, rational functions with finite and/or infinite

*This research was supported in part by the Research Council KU Leuven, projects OT/11/055 (Spectral Properties of Perturbed Normal Matrices and their Applications), OT/10/038 (Multi-Parameter Model Order Reduction and its Applications), CoE EF/05/006 Optimization in Engineering (OPTEC); by the Fund for Scientific Research–Flanders (Belgium) project G034212N (Reestablishing Smoothness for Matrix Manifold Optimization via Resolution of Singularities), by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office, Belgian Network DYSCO (Dynamical Systems, Control, and Optimization); and by a DFG research stipend MA 5852/1-1.

†Department of Computer Science, KU Leuven, 3001 Leuven (Heverlee), Belgium. ({thomas.mach,marc.vanbarel,raf.vandebril}@cs.kuleuven.be).

poles,...), given nodes and weights of the inner product, is typically done by plain matrix operations. By similarity transformations, one transforms the diagonal matrix of eigenvalues (see [11, 16]) to a matrix of certain structure. The eigenvalues and eigenvectors link to the nodes and weights of to the inner product, and the matrix structure connects to the function type and eigenvalue distribution (e.g., Hessenberg vs. plain polynomials, Hermitian tridiagonal vs. polynomials with real eigenvalues, unitary Hessenberg vs. Szegő polynomials, extended Hessenberg without poles vs. Laurent polynomials, extended Hessenberg with poles vs. rational functions).

For a survey of methods on inverse eigenvalue problems, we refer to Chu and Golub [11], Boley and Golub [6], and see also the book of Golub and Meurant [15]. When the structure of the matrix M we are looking for is upper Hessenberg, taking the λ_i all on the real line, leads to the symmetry of this Hessenberg matrix. Hence, it becomes tridiagonal and is nothing else than the Jacobi matrix for the corresponding inner product, i.e., it gives the recurrence coefficients of the corresponding orthogonal polynomials [17]. The discrete least squares interpretations of these methods are presented by Reichel [23] and by Elhay, Golub, and Kautsky [12]. These methods efficiently exploit the tridiagonal structure of the matrix representing the recurrence relations and construct the optimal polynomial fitting in a least squares sense, given the function values in these real points λ_i. Based on the inverse unitary QR algorithm for computing unitary Hessenberg matrices [2], Reichel, Ammar, and Gragg [24] solve the approximation problem when the given function values are taken in points λ_i on the unit circle. Their algorithm is based on computational aspects associated with the family of polynomials orthogonal with respect to an inner product on the unit circle. Such polynomials are known as Szegő polynomials. Faßbender [14] presents an approximation algorithm based on an inverse unitary Hessenberg eigenvalue problem and shows that it is equivalent to computing Szegő polynomials. More properties of the inverse unitary Hessenberg eigenvalue problem are studied by Ammar and He [4].

A generalization of these ideas to vector orthogonal polynomials and to the least squares problems of a more general nature is presented by Bultheel and Van Barel in [9, 27,30]. They developed an updating procedure to compute a sequence of orthonormal polynomial vectors with respect to that inner product where the points λ_i could lie anywhere in the complex plane. Again, if the inner products are prescribed in points on the real axis or on the unit circle, they present variants of the algorithm which are an order of magnitude more efficient. Similarly as in the scalar case, when all λ_i are real, the generalized Hessenberg becomes a banded matrix [1,28], and when all λ_i are on the unit circle, H can be parametrized using block Schur parameters [29]. Also a downdating procedure was developed [31]. For applications of downdating in data analysis, the reader can have a look at [3].

So far, we have only considered polynomial functions. When taking proper rational functions with prescribed poles $y_k \neq \infty$, $k = 1, \ldots, n$, the inverse eigenvalue problem becomes

$$Q^H D_z Q = S + D_y, \tag{1.1}$$

where D_y is the diagonal matrix based on the poles y_k (with an arbitrary value for y_0), and where S has to be lower semiseparable, i.e., all submatrices that can be taken out of the lower triangular part of S have rank at most 1. Also here, when all λ_i are real, S becomes a symmetric semiseparable matrix and when all λ_i lie on the unit circle, S has to be of lower as well as upper semiseparable form [33,34].

In this manuscript we will investigate general, not necessarily proper, rational

functions. We will investigate the structure of the matrix that represents the recurrence coefficients for these sequences of orthogonal rational functions.

The techniques described above can be used in several applications in which polynomial or rational functions play an important role: linear system theory, control theory, system identification [7,22], data fitting [12], (trigonometric) polynomial least squares approximation [23,24], and so on. For a comprehensive overview of orthogonal rational functions, the interested reader can consult [8].

The article is organized as follows. There are two main sections, each discussing an inverse eigenvalue problem. Section 3 discusses the inverse eigenvalue problem for extended Hessenberg matrices: given eigenvalues and a vector of weights, construct via unitary similarity transformations an extended Hessenberg matrix, whose eigenvalues are as defined, and whose orthogonal eigenvectors have as first components the elements of the weight vector. In Section 4 we tackle an inverse eigenvalue problem where given two weight vectors and eigenvalues an extended tridiagonal matrix is constructed, whose eigenvalue decomposition has prescribed eigenvalues and the eigenvectors (not necessarily unitary anymore, but of unit length) have their first components related to the weight vectors. Both these sections are organized alike. First the Krylov subspace, whose orthogonal basis we would like to retrieve is presented and the structure of the matrix of recurrences is deduced. The compact representation of the matrix of recurrences is next presented, and will be used extensively in the algorithm design which relies heavily on basic 2×2 matrix operations. The description of the algorithm itself is subdivided in smaller parts, clearly distinguishing between finite and infinite poles.

We rely on the following notational conventions. Matrices are written as capitals A; the matrix element positioned on the intersection of row i and column j is denoted as a_{ij}. Vectors are typeset in bold: $\mathbf{v}$; the standard basis vectors are the $\mathbf{e}_i$'s. With $\cdot^T$ the transpose is meant; $\cdot^H$ stands for the Hermitian conjugate. Standard `Matlab` notation is used to select submatrices ranging from rows i up to j and columns k up to ℓ: $A(i:j, k:\ell)$; the following shorthand notation $A(k:\ell)$ is used to identify the square submatrix $A(k:\ell, k:\ell)$. With $\mathrm{diag}(\xi_1, \xi_2, \xi_3)$ the diagonal matrix whose diagonal elements take values ξ_1, ξ_2, and ξ_3 is meant.

2. Orthogonal functions and Krylov spaces. There is strong connection between orthogonal polynomials, orthogonal rational functions, and Krylov and rational Krylov bases. In both approaches one constructs either a sequence of polynomials (rational functions) or Krylov vectors, which are then orthogonalized against each other and one stores the recurrence coefficients in a matrix. It's this matrix that one typically wants to retrieve in an inverse eigenvalue problem. More precisely, as an example, we consider the classical orthogonal polynomial – Krylov relation. Given a possibly complex $n \times n$ Hermitian matrix A, the associated Krylov space $\mathcal{K}_\ell(A, \mathbf{e}_1)$ having starting vector $\mathbf{e}_1$ equals the subspace $\mathcal{K}_\ell(A, \mathbf{e}_1) = \mathrm{span}\{\mathbf{e}_1, A\mathbf{e}_1, A^2\mathbf{e}_1, \ldots, A^{\ell-1}\mathbf{e}_1\}$. In the remainder we assume all considered Krylov spaces to be of full dimension, i.e., $\dim \mathcal{K}_\ell(A, \mathbf{e}_1) = \ell$. It is well-known that for an orthogonal Krylov basis $V = [\mathbf{v}_1, \ldots, \mathbf{v}_\ell]$ for $\mathcal{K}_\ell(A, \mathbf{e}_1)$ (where $\ell = 1, 2, \ldots$) the *projected counterpart* $V^H A V$ is of Hessenberg form, i.e., has zeros below the first subdiagonal [16, 41]. Conversely consider the polynomial subspace $\mathcal{P} = \mathrm{span}\{1, x, x^2, x^3, \ldots\}$, and orthogonalize them w.r.t. a discrete inner product $[p_0(x), p_1(x), p_2(x), \ldots]$. Storing the recursions between these orthogonal polynomials in a matrix results in an identical matrix H if the weights of the inner product stem from the eigenvectors's first components and the nodes are the eigenvalues of the matrix. Let $A = Q\Lambda Q^H$ be the eigenvalue decomposition of

the normal matrix A, $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ diagonal, and Q orthogonal. Then the following equalities relate the polynomial orthogonalization procedure to the vector orthogonalization:

$$\langle x^i, x^j \rangle = \sum_{k=1}^{n} w_k^2 \lambda_k^{i+j} = (\mathbf{e}_1^H Q)\Lambda^{i+j}(Q^H \mathbf{e}_1) = \mathbf{e}_1^H A^{i+j} \mathbf{e}_1 = \langle A^i \mathbf{e}_1, A^j \mathbf{e}_1 \rangle.$$

The Krylov subspace does not necessarily need to be initialized with the vector $\mathbf{e}_1$, an arbitrary vector $\mathbf{v}$ is possible, but then the weights determining the inner product need to be adjusted as well. To stress the link even more and to identify possible extensions, consider the polynomials $\varphi_\ell(x) = x^\ell$, with ℓ ranging from 0 to $n-1$. Then we get, for the above subspaces the following:

$$\begin{aligned} \mathcal{K}_\ell(A, \mathbf{e}_1) &= \text{span}\{\varphi_0(A)\mathbf{e}_1, \varphi_1(A)\mathbf{e}_1, \varphi_2(A)\mathbf{e}_1, \ldots\}, \\ \mathcal{P} &= \text{span}\{\varphi_0(x), \varphi_1(x), \varphi_2(x), \ldots\}. \end{aligned}$$

Though we will not elude on this, special chosen φ_i's link classical Krylov spaces to orthogonal polynomials, extended Krylov spaces to Laurent polynomials, classical Krylov spaces (with unitary matrices) to Szegő polynomials, rational Krylov spaces to orthogonal rational functions, and bi-orthogonal rational functions to nonsymmetric Lanczos (this last case requires two Krylov spaces, and two bi-orthogonal sequences of polynomials). In this article, we will stick to matrix inverse eigenvalue problems, where given eigenvalues and weights we reconstruct a matrix whose structure is linked to a particular Krylov subspace; in our setting rational Krylov subspaces and nonsymmetric Krylov subspaces (nonsymmetric Lanczos).

Again in both the functional theory approach and in the matrix setting the matrix of recurrences exhibits a particular structure related to the types of functions under consideration or to the Krylov subspace considered. As we desire to retrieve a matrix of recurrences providing us the recurrence coefficients between the basis vectors of a particular Krylov subspace we first will have to derive this structure. Whereas in the classical orthogonal functions approach one closely examines the relations between the inner products, here we will plainly operate on matrices. For instance, in the classical Krylov space, the matrix will be of Hessenberg form, but depending on the case it can become Hermitian tridiagonal, unitary Hessenberg, nonsymmetric tridiagonal, ...

3. The inverse eigenvalue problem for extended Hessenberg matrices. The inverse problem considered in this section aims at reconstructing the projected counterpart, associated to a rational Krylov space, with predetermined poles, eigenvalues and weights.

Section 3.1 discusses the matrix structure we aim at and the exact problem formulation. Section 3.2 presents a factorization of the desired matrix. In Section 3.3 we elude on the fundamental matrix operations necessary in the algorithm. Section 3.4 solves the inverse eigenvalue problem. We conclude by reporting on some special eigenvalue distributions and their effect on the matrix structure, and present some numerical results in Section 3.6.

3.1. The induced matrix structure & Problem formulation. In the following we draw heavily from [20,37,39]. Given a vector of poles $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_n]$, with $\xi_i \in (\mathbb{C} \cup \{\infty\}) \backslash \Delta(A)$, for $i = 1, \ldots, n$, we define the *rational Krylov* space $\mathcal{K}_\ell^{\text{rat}}$ of

dimension ℓ, with initial vector $\mathbf{v}$ as

$$\mathcal{K}^{\mathrm{rat}}_{\boldsymbol{\xi},\ell}(A,\mathbf{v}) = q_{\ell-1}(A)^{-1}\mathcal{K}_\ell(A,\mathbf{v}), \quad q_{\ell-1}(z) = \prod_{\substack{j=1\\ \xi_j\neq\infty}}^{\ell-1}(z-\xi_j). \tag{3.1}$$

This means that an ℓ-dimensional rational Krylov space is, for instance, of the form

$$\begin{aligned}\mathcal{K}^{\mathrm{rat}}_{\boldsymbol{\xi},\ell}(A,\mathbf{v}) = \mathrm{span}\{&\mathbf{v}, (A-\xi_1 I)^{-1}\mathbf{v}, (A-\xi_2 I)^{-1}(A-\xi_1 I)^{-1}\mathbf{v}, A\mathbf{v},\\ &A^2\mathbf{v}, A^3\mathbf{v}, (A-\xi_6 I)^{-1}(A-\xi_2 I)^{-1}(A-\xi_1 I)^{-1}\mathbf{v}, \ldots\},\end{aligned}$$

given $\boldsymbol{\xi} = [\xi_1, \xi_2, \infty, \infty, \infty, \xi_6, \ldots]$. Each finite pole is linked to a shift-and-invert multiplication, and the infinite poles introduce plain multiplications with powers of A. Thus taking all poles infinite simplifies the problem to the classic one.

To easily track the non-periodic occurrences of finite and infinite poles we introduce the *selection* vector $\mathbf{p}$. This vector only takes values f and i, marking the associated pole as finite or infinite. This distinction is critical, entirely determining the associated matrix structure and more important this vector permits an easy treatment.

In the following table the selection vector, its values, the poles and the associated vectors in the rational Krylov space are presented. For completeness also the orthogonal basis vectors for the rational Krylov space, stored in V, as well as their indices are given. The indices i_j mark transitions from f to i or vice versa when passing from p_{i_j} to p_{i_j+1}. We set $i_1 = 1$ and the trailing i_j equals the matrix' size n. Without loss of generality we assume in the remainder of the text $p_1 = i$, so the subspaces are always initiated with powers of A.

$\mathbf{p}$		p_1	p_2		p_{i_2-1}	p_{i_2}	
p_i		i	i		i	f	
ξ_i		∞	∞	$\cdots$	∞	ξ_{i_2}	$\cdots$
columns of $\mathcal{K}^{\mathrm{rat}}_{\boldsymbol{\xi},\ell}$	$\mathbf{v}$	$A\mathbf{v}$	$A^2\mathbf{v}$		$A^{i_2-1}\mathbf{v}$	$q_{i_2}(A)^{-1}\mathbf{v}$	
columns of V	$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$		$\mathbf{v}_{i_2}$	$\mathbf{v}_{i_2+1}$	

$\mathbf{p}$		p_{i_3-1}	p_{i_3}		p_{i_4-1}	
p_i		f	i		i	
ξ_i	$\cdots$	ξ_{i_3-1}	∞	$\cdots$	∞	
columns of $\mathcal{K}^{\mathrm{rat}}_{\boldsymbol{\xi},\ell}$		$q_{i_3-1}(A)^{-1}\mathbf{v}$	$A^{i_2}\mathbf{v}$		$A^{i_2-i_3+i_4-1}\mathbf{v}$	
columns of V		$\mathbf{v}_{i_3}$	$\mathbf{v}_{i_3+1}$		$\mathbf{v}_{i_4}$	

$\mathbf{p}$	p_{i_4}		p_{i_5-1}	p_{i_5}	
p_i	f		f	i	
p_i	ξ_{i_4}	$\cdots$	ξ_{i_5-1}	∞	$\cdots$
columns of $\mathcal{K}^{\mathrm{rat}}_{\boldsymbol{\xi},\ell}$	$q_{i_4}(A)^{-1}\mathbf{v}$		$q_{i_5-1}(A)^{-1}\mathbf{v}$	$A^{i_2-i_3+i_4}\mathbf{v}$	
columns of V	$\mathbf{v}_{i_4+1}$		$\mathbf{v}_{i_5}$	$\mathbf{v}_{i_5+1}$	

The projected counterpart $Z = V^H AV$ is highly structured. The submatrices $Z(i_j : i_{j+1}+1)$ for odd j, are of Hessenberg form, and the blocks corresponding to even j are of inverse Hessenberg plus diagonal form, with the perturbing diagonal equal to $[0, \xi_{i_j}, \ldots, \xi_{i_{j+1}-1}, 0]$, which is a non-interrupted succession of finite poles

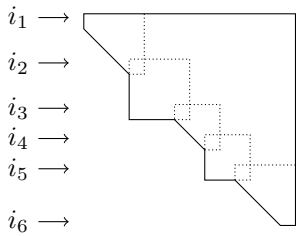


Fig. 3.1: Structure of the projected counterpart.

pre- and postpended with zero, except when $i_j = n$. All the structured diagonal blocks have thus an overlap of a 2×2 submatrix. Figure 3.1 graphically depicts the positioning and overlap of the blocks. In the remainder of this article we will name such matrix an *extended Hessenberg plus diagonal matrix*. This is consistent with the nomenclature of extended Krylov spaces, which lack poles, and solely contain powers and inverses of A, implying thus an empty diagonal term. The matrix Z contains the recurrences for reconstructing the orthogonal column vectors of V which are, as mentioned before, tightly connected to orthogonal rational functions.

Using only infinite poles $V^H AV$ becomes Hessenberg; permitting only zero as a pole $V^H AV$ becomes of extended form as proved in [37]; in [34] it was demonstrated that allowing only finite poles results in an inverse Hessenberg plus diagonal matrix; here, however, a mixture of all these structures is allowed.

INVERSE EIGENVALUE PROBLEM 3.1. *Given a vector* $\mathbf{v} = [v_1, v_2, \ldots, v_n]^T$ *of weights, let* Λ *denote a diagonal matrix with as diagonal values the eigenvalues* $\lambda_1, \lambda_2, \ldots, \lambda_n$. *The vector of poles is* $\boldsymbol{\xi}$, *and* $\mathbf{p}$ *the associated selection vector. The problem to be solved is to efficiently compute the matrix* $V^H \Lambda V$ *(and if required the unitary matrix* V*) such that* $V\mathbf{e}_1 = \mathbf{v}/\|\mathbf{v}\|_2$ *and* $V^H \Lambda V$ *is of extended Hessenberg plus diagonal form, where* D *contains the poles* $\boldsymbol{\xi}$, *and the extended Hessenberg matrix' structure matches the position vector* $\mathbf{p}$.

For finite poles only, this problem was solved in [33, 34]. The introduction of infinite poles is as such slightly more general, but it must be stressed that with some nontrivial modifications the methods proposed in [33, 34] are likely to deal with this problem as well. However, all the theoretical structure predicting proofs needed to be adapted as well to suit this more general setting. Moreover, the methodology and derivation of the algorithm in this article is entirely different, yet yielding the same outcome in case of only finite poles. These deductions do also allow a swift treatment of the more general inverse eigenvalue problem discussed in Section 4.

3.2. Representation. Rather than storing the dense matrix $V^H \Lambda V$ we will factor it by its QR factorization and store the Q factor memory economical by decomposing it as a product of rotations. We will then operate on this rotational factorization.

The key-ingredient is the QR factorization; it is known that the unitary Q factor in the QR factorization of a Hessenberg matrix comprises $n-1$ rotations [10, 40]. More precisely, when G_i denotes the embedding of a rotator in the identity matrix, having its effective part operating on rows i and $i+1$, we have that a Hessenberg's QR factorization can be written as $H = G_1 G_2 G_3 \ldots G_{n-1} R$, in which the Q factor exhibits a *descending* sequence of rotators. The inverse of a Hessenberg matrix, admits

a factorization of form $G_{n-1}G_{n-2}\dots G_2G_1R$, where the factorization of Q is referred to as an *ascending* sequence of rotations; this can be proved easily by utilizing the techniques from Section 3.3.

In general, the projected matrix Z associated to a rational Krylov space can be rewritten as $\tilde{Z} = Z - D$, with D a diagonal matrix with precisely positioned poles on its diagonal, and $\tilde{Z}$ of extended Hessenberg form. The diagonal blocks in $\tilde{Z}$ are thus of Hessenberg or inverse Hessenberg form. Merging the QR factorizations of each of the diagonal blocks results in a *zigzag* shaped ordering of the rotations, where ascending and descending sequences of rotations are taking turns.

Let us clarify by an example how the zigzag shape relates to the diagonal blocks. Take the selection vector as $\mathbf{p} = [\mathsf{i}, \mathsf{i}, \mathsf{i}, \mathsf{f}, \mathsf{f}, \mathsf{f}, \mathsf{f}, \mathsf{i}, \mathsf{i}, \mathsf{i}]$ associated to the vector of poles $\boldsymbol{\xi}$. We have thus $i_1 = 1, i_2 = 4, i_3 = 8, i_4 = 11$. We get for starting vector $\mathbf{v}$ that $\mathcal{K}^{\mathrm{rat}}_{\boldsymbol{\xi},11}(A, \mathbf{v}) = \{\mathbf{v}, A\mathbf{v}, A^2\mathbf{v}, A^3\mathbf{v}, q_4(A)^{-1}\mathbf{v}, q_5(A)^{-1}\mathbf{v}, q_6(A)^{-1}\mathbf{v}, q_7(A)^{-1}\mathbf{v}, A^4\mathbf{v}, A^5\mathbf{v}, A^6\mathbf{v}\}$. The associated projected counterpart $\tilde{Z} = Z - D$ obeys the structure visualized in Figure 3.2. The brackets with arrows denote rotations acting on the rows marked with the arrows. The crosses stand for possibly non-zero entries of the matrix. The three overlapping diagonal blocks are Hessenberg, inverse Hessenberg, and

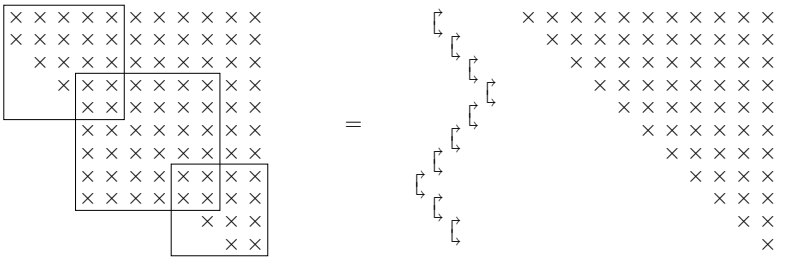


Fig. 3.2: QR factorization of an extended Hessenberg matrix.

again Hessenberg and they are emphasized and put in a square on the left. The right scheme reveals the graphical QR factorization of this matrix. To retrieve the QR factorization of the matrix $\tilde{Z}$ first the inverse Hessenberg blocks are factored with ascending sequences of rotations (counting from top to bottom we regard rotators 5 up to 8), next the Hessenberg blocks are factored by descending sequences of rotations (rotations 1 up to 4, and 9 up to 10).

A rigorous analysis of the ordering of the rotations furnishes us the following rule. Again each rotation G_i is expected to operate on rows i and $i+1$. Whenever the ith component p_i of the selection vector p_i equals i, rotation G_i is positioned to the left of rotator G_{i+1}, value $p_i = \mathsf{f}$ signifies that G_i is positioned to the right of G_{i+1}.

The structure we will try to retrieve in the inverse problem is $Z = QR + D$, with D the diagonal with correctly positioned poles, and QR the QR factorization of the matrix $\tilde{Z}$, with Q factored in rotations according to the selection vector $\mathbf{p}$.

3.3. Manipulating rotations. To work effortlessly with rotators, three basic operations are necessary: the *fusion*, the *turnover*, and the *pass through* action.

The fusion is the most elementary operation: Two successive rotations operating on identical rows can be united in a single rotation. Figure 3.3(a) illustrates this.

The turnover action reshuffles three successive rotations: let F, G, H be three rotators affecting rows $1-2$, $2-3$, and $1-2$ respectively. Then it is always possible

Algorithm 1: Extended Hessenberg Inverse Eigenvalue Problem.

Input: $\mathbf{\Lambda} \in \mathbb{C}^n$, $\mathbf{v} \in \mathbb{C}^n$, ξ
Output: $H = QR + D \in \mathbb{C}^{n\times n}$, V *(if required)*
$R = \text{diag}(\mathbf{\Lambda})$; $D = \text{zeros}(n, n)$; $Q = \text{eye}(n, n)$;
$V = \text{eye}(n, n)$ *(if required)*;
Compute G that zeroes v_n;
Apply the similarity transformation on R: GRG^H;
Update the QR decomposition and the poles in $QR + D$;
for $k = n - 1 : 2$ **do**
  Compute G that zeroes v_k;
  Apply the similarity transformation and generate the bulge;
  Chase the bulge as described in Section 3.4,
  thereby update the poles and V *(if required)*;
end

to refactor their product again in three rotations V, W, and X, operating now on rows $2-3$, $1-2$, and $2-3$ so that $FGH = VWX$. Figure 3.3(b) sheds some light on this change and reshuffle.

In the passing through operation we have a pattern of rotations gathered in the matrix $\tilde{Q}$ positioned to the right of an upper triangular matrix $\tilde{R}$. Applying the leftmost rotation of $\tilde{Q}$ on the matrix $\tilde{R}$ (assume it acts on columns i and $i+1$) creates a bulge in position $(i+1, i)$. This bulge can be eliminated by a rotation on the left, acting on rows i and $i+1$. In this manner one can pass one by one the rotations in $\tilde{Q}$ through the upper triangular matrix and obtain a factorization QR, with Q's shape identical to the one of $\tilde{Q}$. The transition of all rotators on the right to the left is named the passing through operation. Of course a similar operation from left to right is also feasible. Figure 3.3(c) depicts this.

(a) Fusion (b) Turnover (c) Passing through

Fig. 3.3: Three basic operations on rotators.

3.4. Algorithmic solution. To tackle Problem 3.1, we rely on the results from Section 3.2 and do not compute the matrix $V^H \Lambda V$ directly, but in factored form. A pseudo-code of the algorithm is already depicted in Algorithm 1 which is detailed in this section. We initiate the discussion by dealing with the cases with only infinite and only finite poles followed by a discussion on their integration.

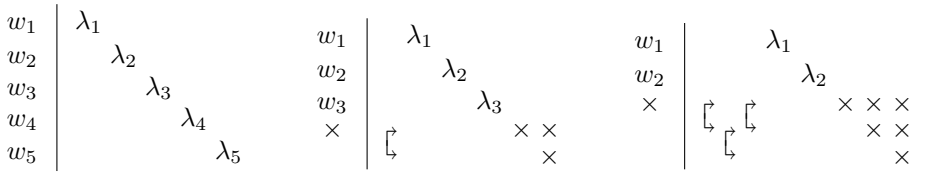
3.4.1. Only poles at infinity. In this section the classical inverse eigenvalue problem to retrieve a Hessenberg matrix, given its eigenvalues and first vector of the unitary matrix V is considered. Though well-known [33,34], the description with rotators is new and enhances the comprehension of the generic algorithm. The algorithm

will be presented in a graphical manner, so we wish to retrieve the QR factorization of the Hessenberg matrix, consequently configured as a descending sequence of rotators.

Figure 3.4(a) depicts the initial setup for an example of dimension 5. A vertical line separates the vector of weights from the diagonal matrix of eigenvalues. Each step $i = 1, \ldots, n-1$ of the algorithm the $(n-i+1)$st element of the current weight vector is zeroed by a rotator acting on rows $n-i$ and $n-i+1$. This rotation determines the similarity to be executed on the associated matrix. Following this first similarity $i-1$ other similarities (with rotations) are needed to remove the fill-in generated by the first similarity. As result we obtain the Hessenberg matrix in factored form.

Let us elude more on this by examining the 5×5 example. We assume that after a similarity the rotations on the right will always be passed through the upper triangular matrix immediately, making them thus appear on the left. Apart from the elements agreeing to those in the first figure, we will simply denote the matrix and weight elements by a $\times$.

Figure 3.4(b) depicts the result after annihilating the first weight, and performing the coupled similarity; the rotation positioned to the right of the matrix is first passed through the upper triangular (diagonal) matrix after which it is fused with the left positioned rotation. In Figure 3.4(c) the first similarity transformation of step 2 is executed; it is plainly visible that there is one additional rotation. To remove this rotation, first a turnover is needed resulting in Figure 3.4(d). The outer left rotation determines the forthcoming similarity, such that the rotation itself vanishes after executing it; the similarity also introduces a rotator to the right, which after passing through the upper triangular matrix can be fused with the bottom rotator, providing us the result of step 2 in Figure 3.4(e). Crucial in the latter similarity transformation, designed to chase the perturbing rotation, is that the zeros of the weight vector, after operating on the left, remain intact.



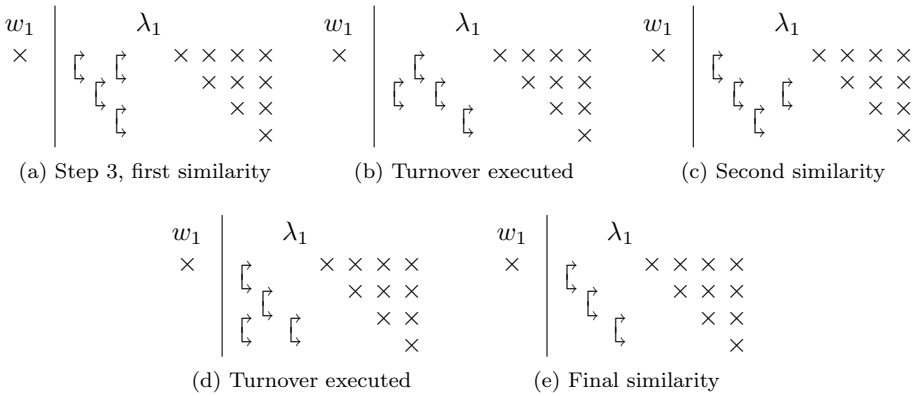
(a) Initial setting (b) Entire step 1 (c) Step 2, first similarity

(d) Step 2, turnover executed (e) Step 2, final similarity

Fig. 3.4: Inverse eigenvalue problem having only poles at infinity (steps 1 and 2).

Figure 3.5 shows the flow of step 3. After the first similarity (Figure 3.5(a)), again an undesired rotation is introduced. The turnover operation moves the unwanted rotation to the left and deposits it one row lower (Figure 3.5(b)). A second similarity removes the outer left rotation and introduces another, still undesired rotation, on the right (Figure 3.5(c)). Important is that the sparsity of the weight vector is maintained.

Another turnover brings this rotation to the left, once more positioned a row lower (Figure 3.5(d)), which is removed next by a similarity (Figure 3.5(e)).



(a) Step 3, first similarity (b) Turnover executed (c) Second similarity

(d) Turnover executed (e) Final similarity

Fig. 3.5: Inverse eigenvalue problem having only poles at infinity (step 3).

Continuing this procedure results in a Hessenberg matrix. Accumulating all rotations in a matrix V^H. We have, by construction, that $V^H \Lambda V$ is of Hessenberg form and $V^H \mathbf{v} = \alpha \|\mathbf{v}\|_2 \mathbf{e}_1$, with $|\alpha| = 1$, satisfying the requirements. The last step is the multiplication of the first column of V with $\overline{\alpha}$ and the first column and row of H with α. This does not effect the structure of H and the problem is thus solved.

3.4.2. Only finite poles. A solution to this problem was already proposed in [33, 34]. Though the approach presented in this section is theoretically equivalent, the onset and accordingly the entire description differs significantly. This differing approach provides additional insight and enables us to smoothly combine this and the algorithm of Section 3.4.1 to tackle the general setting.

The introduction of finite poles forces us to consider the more general factorization with the perturbing matrix D containing the poles. The basic idea is identical to Section 3.4.1. In each step the rotation determining the first similarity is designed to zero out an element in the weight vector followed by a succession of similarities to chase the remaining uninvited rotation. There are two crucial differences: whereas in the Hessenberg case the outer left rotators were selected to execute a similarity with, here the outer right ones will be taken. Another difference is the introduction of the poles, which require special handling. In the end we desire a factorization $QR + D$, with Q factored as an ascending sequence of rotations, R upper triangular and $D = \text{diag}(0, \xi_1, \ldots, \xi_{n-1})$.

Figure 3.6 exemplifies the flow of the algorithm for a 5×5 matrix. Initially (Figure 3.6(a)) we start with the diagonal matrix $\Lambda = Q_0 R_0$ plus a zero diagonal matrix D_0. The QR factorization of Λ will gradually be transformed to a QR factorization of desired form, whereas the matrix D_0 will be modified to position the poles on the diagonal. Executing the initial similarity of step 1 proceeds identical as in Section 3.4.1 Figure 3.6(b). After this, one needs to fiddle with both terms to introduce the first pole in the zero diagonal matrix, the altered elements in the left term are typeset bold in Figure 3.6(c), and the first pole is introduced. After step 1 we have $Q_1 R_1 + D_1$, where Q_1 contains one rotator, R_1 is block diagonal, with one diagonal, and one upper triangular block, and D_1 embodies the first pole. Step 2 kicks off as before. After

the similarity, passing through, and turnover we arrive at $\tilde{Q}_2\tilde{R}_2 + D_1$, visualized in Figure 3.6(d). It is relevant to mention that the similarity transformation did not alter the diagonal matrix D_1, its action was restricted to a part with equal diagonal values. Stated already before, the next similarity transformation, should be designed to remove the outer right rotator of $\tilde{Q}_2$. First, however, again a rewriting of the sum is required, as otherwise the similarity would destroy the requested diagonal structure of D_1. The diagonal is altered to $\tilde{D}_2 = \text{diag}(0, 0, 0, \xi_1, \xi_1)$, ensuing that a new QR factorization of the resulting modified left term $\tilde{Q}_2\tilde{R}_2 - (\tilde{D}_2 - D_1)$ is preferred. We get

$$\tilde{Q}_2(\tilde{R}_2 - \tilde{Q}_2^H(\tilde{D}_2 - D_1)) = \tilde{Q}_2\hat{R}_2, \tag{3.2}$$

where the term $\tilde{Q}_2^H(\tilde{D}_2 - D_1)$ has only the three trailing elements in the penultimate column different from zero. Adding these elements to the upper triangular matrix $\tilde{R}_2$ gives us $\hat{R}_2$, with a graphical representation of (3.2) in the left term of Figure 3.6(e), displaying the affected elements in bold. There are now a troubling rotator and bulge simultaneously in the factorization $\tilde{Q}_2\hat{R}_2$. Fortunately both of them can be removed at once by a similarity. To pass the outer right rotator of $\tilde{Q}_2$, say $\hat{G}$, through $\hat{R}_2$ one applies it on $\hat{R}_2$, providentially not creating a bulge, as it already exists. To remove this bulge one first passes this rotation to the right of the factorization: $\hat{G}\hat{R}_2 = \breve{R}_2 G$. This rotator G on the right now determines the next similarity transformation to be executed. The similarity does not destroy the structure of $\tilde{D}_2$ as the active part of the similarity operates on a scalar multiple of the identity in $\tilde{D}_2$. The newly appearing rotation G^H on the left can be fused with the left rotation in $\tilde{Q}_2$, we get $G^H\tilde{Q}_2\hat{G}^H = Q_2$ and $Q_2\breve{R}_2 + \tilde{D}_2$. We end by modifying $\breve{R}_2$, and $\tilde{D}_2$ to obtain $Q_2R_2 + D_2$, with $D_2 = \text{diag}(0, 0, 0, \xi_1, \xi_2)$ as depicted in Figure 3.6(f).

In Figure 3.7 step 3 is summarized. Besides the fact that more chasing steps are now required, the ideas remain unaltered. Figures 3.7(a) and (b) show the result after the first similarity and turnover, and the result of modifying the diagonal of poles. The first similarity to chase the perturbations away and the following up rewriting of the sum are depicted in Figures 3.7(b) and (c). Next another similarity and rewriting needs to be executed almost identical to Step 2.

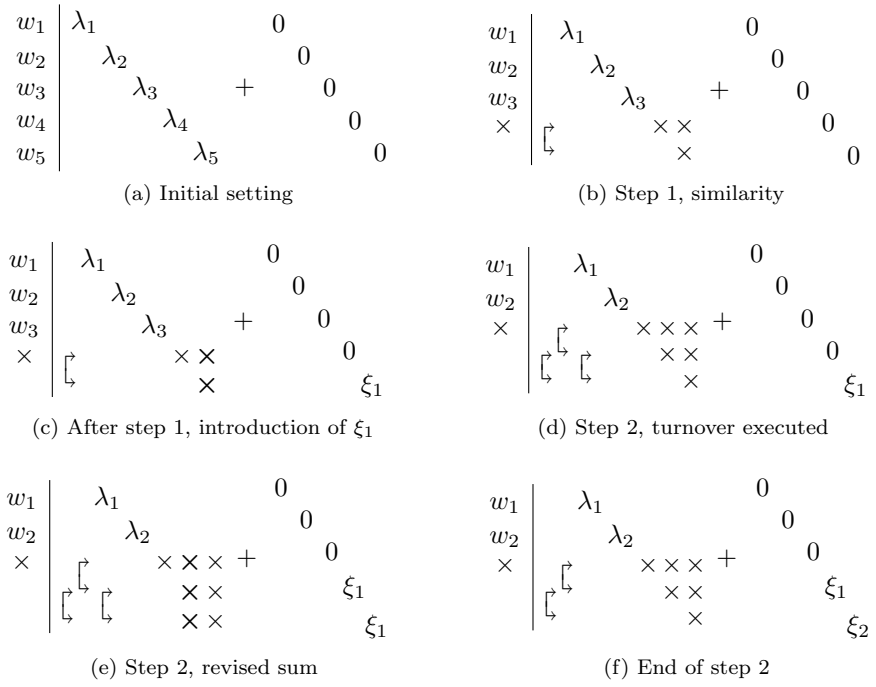
Again it is easily verified that synthesizing all rotations executed results in a matrix V^H, which satisfies the beforehand stated conditions.

REMARK 3.2. *The algorithm proposed in this section admits a slightly faster implementation (see [33, 34]) when one stores each of the diagonal inverse Hessenberg blocks appearing in the extended Hessenberg matrix. Each of these blocks must then be written as an outer product of two vectors. This compact writing allows the modifications to the diagonal of poles in each chasing step to be performed more efficiently.*

3.4.3. General case. It remains to integrate both the algorithms for finite and infinite poles by illustrating how to enforce the transitions from left to right and vice versa in the zigzag shape.

A study of both algorithms presented so far reveals, that once the chasing is initiated in the descending or ascending direction that all similarities are determined by either the outer left or outer right rotations. The remaining rotations are captured in the descending or ascending sequence. This statement holds for all except the final rotation (see Figures 3.7(c) and 3.7(d)) this rotation is not blocked by other rotations, and so one has the freedom to remove either the right or the left rotation.

Summarizing, in each step a new rotator on top is introduced along the descending or ascending line. Once reached the bottom (assume we can pass bends), one has the



(a) Initial setting

(b) Step 1, similarity

(c) After step 1, introduction of ξ_1

(d) Step 2, turnover executed

(e) Step 2, revised sum

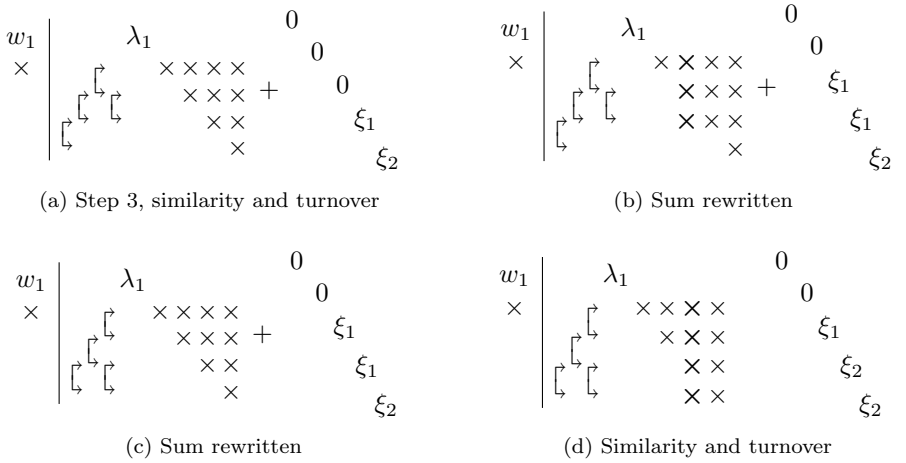
(f) End of step 2

Fig. 3.6: Inverse eigenvalue problem having only finite poles (steps 1 and 2).

liberty to remove either the left or right rotator and as such change the direction of the pattern. Each step (removal of an element in the weight vector) can thus be interpreted as pushing up the existing pattern of rotations, with flexibility of putting the new bottom rotation to the left or to the right of the existing ones. We will consecutively address three subproblems: how to pass a bend in the shape, how to carry out the transitions from finite to infinite poles and vice versa.

Passing a bend during chasing. We commence with tackling *the bend problem*. We assume that so far the chasing went smoothly and we arrived at the bend displayed in Figure 3.8(a) exhibiting part of a pattern with a bend. The matrix continues thus on the upper left and lower right corner. The vector of weights is no longer depicted. After the similarity shown in Figure 3.8(a) the turnover is performed leading to Figure 3.8(b). This figure illustrates that only one option, namely removal of the right rotator remains; recall, however, that before this point a descending sequence was encountered, so all similarities were determined by rotators positioned on the left of the sequence. From this point on we can proceed identically as in Section 3.4.2. To illustrate that this bend will move up one position Figures 3.8(c) and 3.8(d) are included illustrating the modification of both terms before the similarity, and the result of the next similarity. Comparing Figures 3.8(a) and 3.8(d) plainly reveals the upward move of the pattern.

The other bend from finite to infinite is more involved. In Section 3.4 we deliberately did not specify the structure of the matrix D. Whereas one would tacitly assume the infinite poles to be linked to zeros in D, our representation does not allow an efficient algorithmic design when allowing this. Instead we will repeat the previously



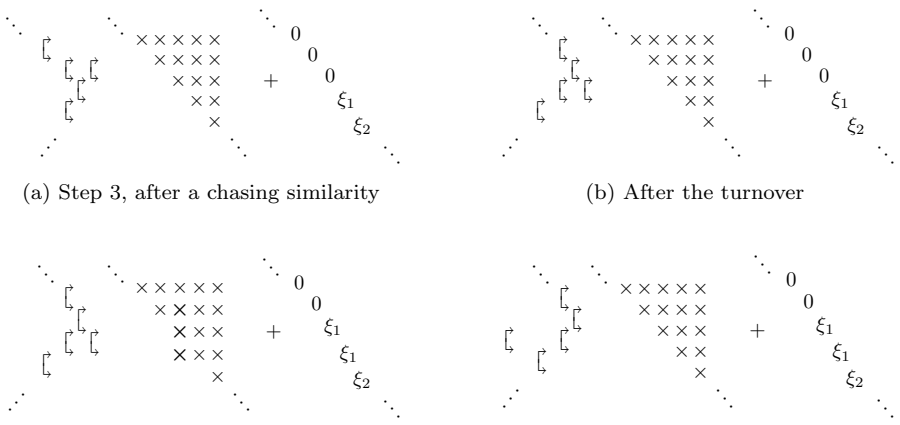
(a) Step 3, similarity and turnover

(b) Sum rewritten

(c) Sum rewritten

(d) Similarity and turnover

Fig. 3.7: Inverse eigenvalue problem having only finite poles (step 3).



(a) Step 3, after a chasing similarity

(b) After the turnover

(c) Sum rewritten

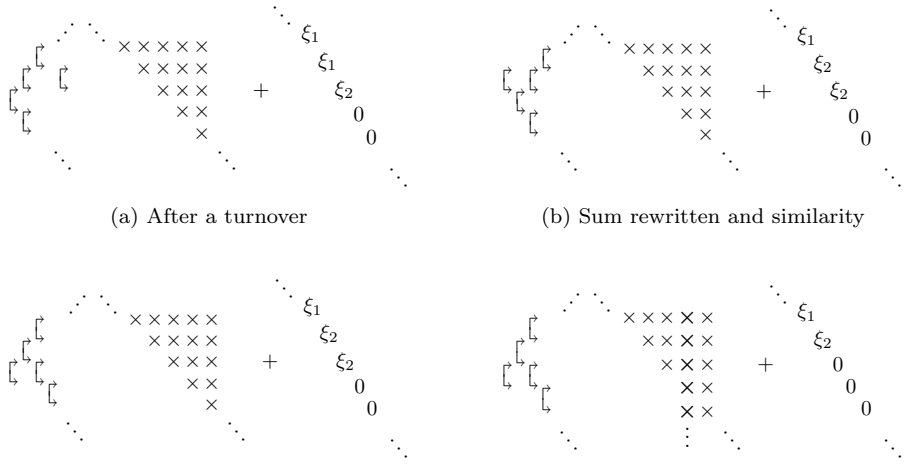
(d) Similarity and turnover

Fig. 3.8: Passing a bend in the shape.

positioned pole in D as long as the actual associated pole remains infinite.

Let us clarify the problem and elegance of this solution further. Let us retain the straightforward idea of having infinite poles linked to zeros in D. Assume at a certain point we have completed a turnover and we arrive in Figure 3.9(a). Rewriting of the sum (modifying the second ξ_1 into ξ_2) followed by a similarity give us Figure 3.9(b). After the next turnover, in which we pass the bend (Figure 3.9(c)) the problems become clear. To modify the trailing ξ_2 in 0 a rewriting of the sum is required, however, this would create a downward spike of undesired non-zeros up to the next bend. This spike cannot be removed, so we have to prevent it from the start by, e.g., keeping the pole ξ_2 instead of trying to make it zero. Since the vectors $\{v, (A-\sigma I)v\}$

span the same space as $\{v, Av\}$, keeping the pole ξ_2 does not affect the infinite poles. Let us see what happens if ξ_2 were repeated. We proceed almost identical, as we



(a) After a turnover

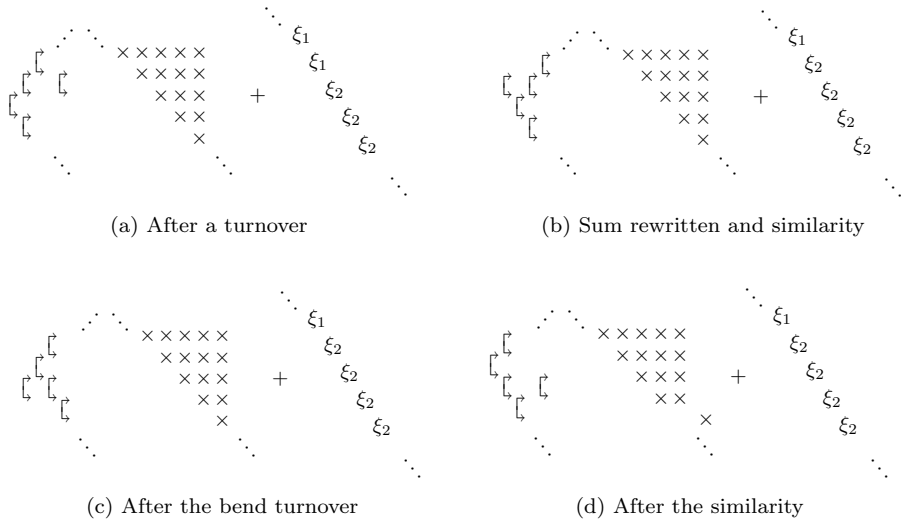
(b) Sum rewritten and similarity

(c) After the turnover

(d) Fill in after rewriting

Fig. 3.9: Possible problems in passing a bend from finite to infinite poles.

assume again at a certain point to have completed a turnover (Figure 3.9(a)). Again the sum is modified and followed by a similarity (Figure 3.9(b)). The last two Figures, Figures 3.10(c) and (d), show again that the pattern moves up one position and no special rewriting of the sum is required anymore. We can retain our efficient QR representation.



(a) After a turnover

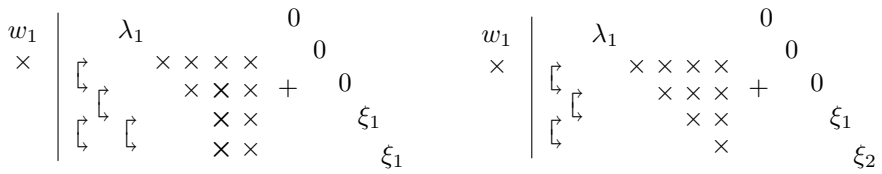
(b) Sum rewritten and similarity

(c) After the bend turnover

(d) After the similarity

Fig. 3.10: Passing a bend from finite to infinite poles in the shape.

Interchanging the pole type: from infinite to finite and vice versa. Next we illustrate the transition from *infinite to finite* poles, on a 5×5 example where the first two rotations were put descendingly. Suppose we arrived at the situation depicted in Figure 3.4(e), and then after some chasing steps we arrive in Figure 3.5(d). To continue the design of a descending sequence one just follows Figure 3.5. Here, however we desire to change the direction. Before the similarity the diagonal matrix (zero in this case) needs to be introduced. Figure 3.11(a) depicts the novel situation. After the similarity determined by the right rotator and some rewriting to introduce pole ξ_2 we get Figure 3.11(b). The onset is now made and a combination of the chasing



(a) Altering the sum, to introduce the switch (b) Similarity determined by right rotator

Fig. 3.11: Transition from infinite to finite poles.

in the descending sense, passing a bend, and chasing in the ascending sense, one can pursue the introduction of new rotations to prolong the current ascending sequence.

The transition from *finite to infinite* poles acts in accordance with the previously described transition. The details are left to the reader, who should not forgot the repeat the final finite pole for being able to effectively pass bends.

Integrating Sections 3.4.2 and 3.4.1 with the last-mentioned transition actions bears us the desired algorithm to solve the inverse eigenvalue problem.

3.5. Particular configurations of eigenvalues. In this article we refrain from adapting and optimizing our algorithm for particular configurations of eigenvalues. We do present the effect on the matrix structure for some classics below. More general cases, e.g., eigenvalues on curves, can be found in [13,19,21] and the references therein.

All eigenvalues located on the real line implies symmetry, i.e., $Z^H = V^H \Lambda^H V = V^H \Lambda V = Z$. Hence the Hessenberg blocks become tridiagonal and the inverse Hessenberg plus diagonal parts become semiseparable plus diagonal blocks. Since we have $Z = V^H \Lambda V$ and $Z^H = V^H \Lambda^H V = Z$. It is evident that exploitation of this structure is compulsory for the development of an efficient algorithm. If we choose the eigenvalues on an arbitrary line in the complex plane we loose the symmetry but still get a tridiagonal matrix. Since there is a $\widetilde{\Lambda} \subset \mathbb{R}$ with $\Lambda = \alpha\widetilde{\Lambda} + \beta I$ and the sparsity pattern of Z, written as

$$Z = V^H \Lambda V = \alpha V^H \widetilde{\Lambda} V + \beta V^H V,$$

is not affected by multiplying with α or by shifting with β. Also the semiseparable structure of the upper and lower triangular parts is not affected.

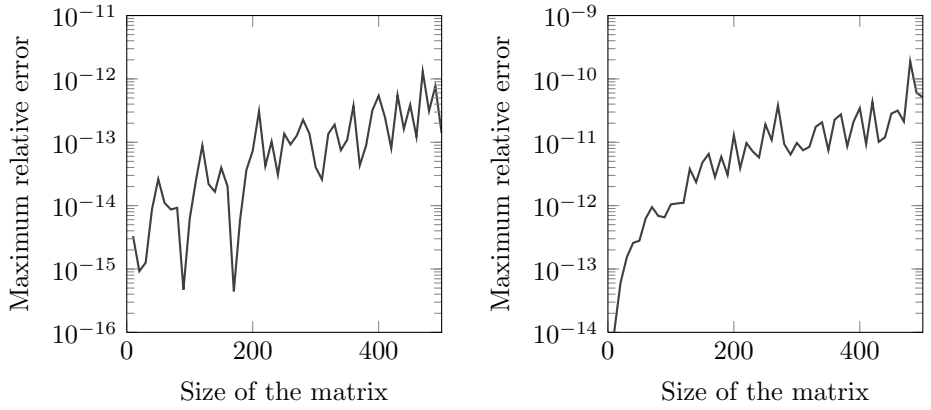
All eigenvalues situated on the unit circle, thus of modulus 1, signifies that the associated matrix is unitary. A unitary Hessenberg matrix has the attractive property that his lower triangular part is sparse and its upper triangular part is rank structured. Allowing only infinite and zero poles one can prove that the associated matrix of recurrences can be represented by solely a zigzag sequence of rotations. Considering

the dense matrix, one acquires alternating and overlapping diagonal blocks of upper Hessenberg and lower Hessenberg form.

If all eigenvalues lie on an arbitrary circle, then we can again shift and scale the problem so that the eigenvalues lie on the unit circle. The scaling does not change the rank structure nor the sparsity pattern of the resulting matrix. But the shifting changes the diagonal and so Z has only a rank structure in the strict lower resp. upper triangular part.

3.6. Numerical experiments. We have tested our algorithm with sets of random eigenvalues in the complex plane ($\lambda = \texttt{rand}(n,1)+i*\texttt{rand}(n,1)$), random complex weights (generated identical to the eigenvalues) and a random selection vector, where we change from infinite to finite and vice versa with a probability of 30%. We ran for each matrix dimension five tests. The maximum relative error is the maximum over all five tests. After having computed the projected counterpart, we recomputed its eigenvalue decomposition providing us eigenvalues $\tilde{\lambda}_i$ and weights $\tilde{w}_i$. For each run the relative error on the eigenvalues was computed as $\max_i |\tilde{\lambda}_i - \lambda_i|/|\lambda_i|$, the relative error of the weights was computed identically.

Figures 3.12(a) and 3.12(b) show that the computed matrix have the wanted eigenvalues and the wanted weights. Figure 3.13 shows that the algorithm is of cubic complexity. The computation time is averaged over the five tests. The experiments were conducted in `Matlab` on an Intel®Xeon®CPU E5645 with 2.40 GHz.



(a) Relative accuracy of the eigenvalues

(b) Accuracy of the weights

Fig. 3.12: Relative accuracy.

4. The inverse eigenvalue problem for extended tridiagonal matrices. In this section we will reconstruct the projected counterpart, linked to the nonsymmetric Lanczos algorithm, with predetermined poles, eigenvalues and weights. This projected counterpart contains the recurrences between biorthogonal sequences of rational functions.

In Section 4.1 we prove that the projected counterpart will be of extended tridiagonal form. A compact factorization of this matrix is essential for the development of an efficient algorithm and is presented in Section 4.2. As we loosen the orthonormality constraint on the basis vectors, we cannot rely solely on unitary operations anymore

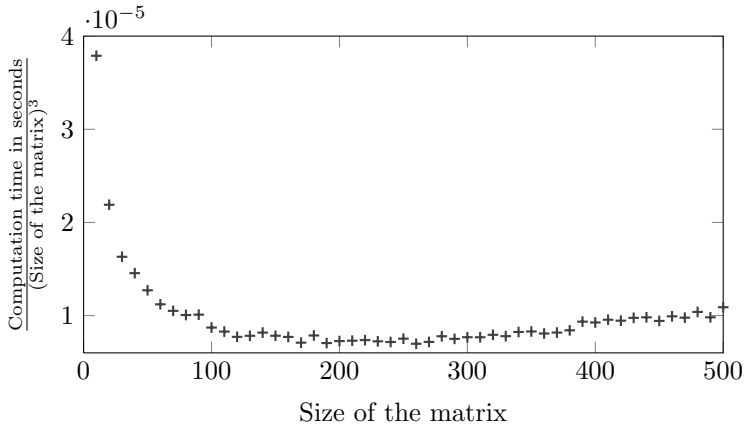


Fig. 3.13: Computational complexity.

and also eliminators will be used. Section 4.3 discusses the essential operations for the algorithm design. In Section 4.4 the algorithm is presented, followed by Numerical experiments in Section 4.5.

4.1. Induced matrix structure & Problem formulation. Detailed knowledge of the structure of the matrix capturing the recurrence coefficients is essential when trying to reconstruct it. To derive this structure we draw heavily from Section 3.1.

Given two vectors of poles $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_n]$, and $\boldsymbol{\chi} = [\chi_1, \chi_2, \ldots, \chi_n]$ with $\xi_i, \chi_i \in \{0, \infty\}$, for $i = 1, \ldots, n$, the following two extended Krylov spaces are considered $\mathcal{K}^{\text{rat}}_{\boldsymbol{\xi},\ell}(A, \mathbf{v})$ and $\mathcal{K}^{\text{rat}}_{\boldsymbol{\chi},\ell}(A^H, \mathbf{w})$ for two vectors of weigths, and under the additional assumption that $\mathbf{w}^H\mathbf{v} = 1$. In this article we exclude pure finite poles.

The upcoming deductions differ significantly from the conventional manner of iteratively constructing the bases W and V. This alternative matrix view enables us, however, to reuse the results from Section 3.1 for deducing the structure of W^HAV swiftly and comprehensibly. Assume, we have constructed $\hat{V}$ and $\hat{W}$ as orthogonal bases for both extended Krylov spaces. We know that $\hat{K}$ and $\hat{Z}$ satisfy

$$A\hat{V} = \hat{V}\hat{Z} \quad \text{and} \quad A^H\hat{W} = \hat{W}\hat{K} \tag{4.1}$$

obeying the extended Hessenberg structure presented in Section 3.1. Hence both matrices $\hat{K}$ and $\hat{Z}$ have thus alternating and overlapping diagonal blocks of Hessenberg and inverse Hessenberg form, imposed by the setting of the poles in $\boldsymbol{\xi}$ and $\boldsymbol{\chi}$.

The actual desired bases V and W need to span the same intermediately generated Krylov spaces as $\hat{V}$ and $\hat{W}$ and must be biorthogonal $W^HV = I$. These matrices can be retrieved as $V = \hat{V}(DU)^{-1}$ and $W^H = L^{-1}\hat{W}^H$, where $\hat{W}^H\hat{V} = LDU$ is an LDU decomposition[1], which means L is unit lower triangular, U unit upper triangular, and D diagonal. Plugging this decomposition in (4.1), we have that

$$AV = A\hat{V}(DU)^{-1} = \hat{V}(DU)^{-1}(DU)\hat{Z}(DU)^{-1} = V\left((DU)\hat{Z}(DU)^{-1}\right) = VZ,$$
$$A^HW = A^H\hat{W}L^{-H} = \hat{W}L^{-H}L^H\hat{K}L^{-H} = W(L^H\hat{K}L^{-H}) = WK.$$

[1]Obviously an LDU factorization need not exist. In this case this procedure, just as the classical iterative algorithm, will break down.

We are fortunate now as multiplications with upper triangular matrices to the left and to the right of a matrix do not have an impact on the lower triangular structure. As a result $\hat{Z}$ and Z, and $\hat{K}$ and K share their structure, i.e., the Hessenberg and diagonally perturbed inverse Hessenberg blocks are positioned identically, even the diagonal elements remain untouched. Moreover, by construction we have now that $Z = K^H$ and as such we have the structure for the lower as well as the upper triangular part.

Whereas in the original case as in Section 3 structure (letting slip the configuration of eigenvalues, see Section 3.5) was only imposed below the diagonal, we now have structure above as well as below the diagonal. Note, however, that these two structures are unrelated. Both are only and strictly connected to the individual independently chosen Krylov spaces. Figure 4.1 reveals a possible configuration, where obviously upper and lower triangular structures are uncorrelated. It might seem unnatural that

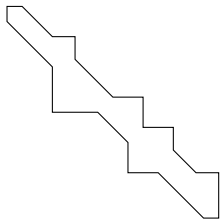


Fig. 4.1: Possible structure in the nonsymmetric Lanczos case.

the upper part apparently has two successive perturbed inverse Hessenberg blocks. Taking into account an overlap of 2×2 blocks, this can be enforced by a single infinite pole in the sequence. To match our previous nomenclature, though slightly misleading as the matrix is not tridiagonal, we will address this type of matrix anyway as an *extended tridiagonal matrix*[2].

Inverse Eigenvalue Problem 4.1. *Consider the vectors* $\mathbf{w} = [w_1, w_2, \ldots, w_n]^T$ *and* $\mathbf{v} = [v_1, v_2, \ldots, v_n]^T$ *containing the weights and let* $\mathbf{w}^H\mathbf{v} = 1$. *Let* Λ *be the diagonal matrix of eigenvalues. The poles for the Krylov spaces with* A *and* A^H *are stored in the vectors* $\boldsymbol{\xi}$ *and* $\boldsymbol{\chi}$ *respectively*[3]. *The inverse problem involves the construction of the matrix* $W^H\Lambda W^{-H} = W^H\Lambda V = V^{-1}\Lambda V$ *(and if desired the matrices* V *and* W*) satisfying the structural constraints as imposed by the vectors of poles as presented in Section 4.1. Moreover the two matrices* V *and* W *have their first vectors equaling* $\mathbf{v}$ *and* $\mathbf{w}$*, up to a scalar, and they are biorthogonal. This means* $W\mathbf{e}_1 = \omega\,\mathbf{w}$ *and* $V\mathbf{e}_1 = \nu\,\mathbf{v}$*, with* ω *and* ν *two scalars; the biorthogonality implies* $W^HV = 1$.

The algorithm presented in this article to solve this problem can suffer from numerical instabilities. Furthermore, like in the classical nonsymmetric Lanczos, breakdowns and near breakdowns can occur. Look-ahead is a possible solution; this is the subject of future research.

4.2. Representation. In the first inverse eigenvalue problem, most of the operations involved manipulating rotations. As we step away from unitarity here, we must also refrain from relying only on manipulations involving rotators.

[2] Every extended tridiagonal matrix is quasiseparable, but the other statement does not hold: a quasiseparable matrix is not necessarily of extended tridiagonal form.

[3] The problem and solution presented here are not yet capable of dealing with finite, nonzero poles. This is the subject to future investigations.

Instead we will use now *Gaussian elimination matrices* zeroing only one element at once, and also changing only one row or column in doing so. To have a short name consistent with *rotator*, we will address these matrices as *eliminators*. Contrary to the rotators, there are two types of eliminators having either

$$\begin{bmatrix} 1 & 0 \\ \times & 1 \end{bmatrix} \qquad \text{or} \qquad \begin{bmatrix} 1 & \times \\ 0 & 1 \end{bmatrix}$$

embedded in the identity matrix, named *lower* resp. *upper eliminators* as they are lower resp. upper triangular. As before we will use a bracket to represent such an eliminator. This bracket will only have one arrow pointing to the row containing the entry $\times$. Graphically we thus get

$$\begin{bmatrix} \\ \hookrightarrow \end{bmatrix} = \begin{bmatrix} 1 & \\ \times & 1 \end{bmatrix} \qquad \text{and} \qquad \begin{bmatrix} \rightarrow \\ \end{bmatrix} = \begin{bmatrix} 1 & \times \\ & 1 \end{bmatrix}.$$

When executing an eliminator-matrix product the bracket also allows another interpretation: in the resulting matrix the row pointed to will change by adding a multiple of the row marked by the bracket's other leg to it. Again only eliminators acting on successive rows are allowed, this means that one can only add a multiple of row i to row $i+1$ or row $i-1$ when executing an eliminator-matrix product. The lower and upper triangular eliminators are notated as L_i and U_i respectively, where the i indicates the row affected under the multiplication, this means that only $L_2, \ldots, L_n$ and $U_1, \ldots, U_{n-1}$ are used.

Furthermore, we will no longer represent our matrix by the QR factorization consisting of a single zigzag shaped series of rotations and an upper triangular. We replace it by an LDU factorization, where both L and U are zigzag shaped products of $n-1$ eliminators: $L = L_{p(2)} \cdots L_{p(n)}$ and $U = U_{q(1)} \cdots U_{q(n-1)}$, where p and q are permutations in $\{2, \ldots, n\}$ and $\{1, \ldots, n-1\}$ respectively. The matrix D is diagonal. For instance, let T be a tridiagonal matrix with LDU factorization $T = LDU$. One can show that the matrices L and U can be factored in a product of eliminators like

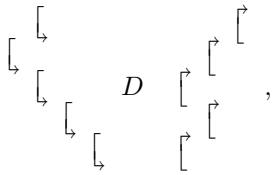
$$\begin{bmatrix} 1 & & & & \\ \times & 1 & & & \\ & \times & 1 & & \\ & & \times & 1 & \\ & & & \times & 1 \end{bmatrix} D \begin{bmatrix} 1 & \times & & & \\ & 1 & \times & & \\ & & 1 & \times & \\ & & & 1 & \times \\ & & & & 1 \end{bmatrix} = \begin{bmatrix} \\ \hookrightarrow \end{bmatrix} \begin{bmatrix} \\ \hookrightarrow \end{bmatrix} \begin{bmatrix} \\ \hookrightarrow \end{bmatrix} \begin{bmatrix} \\ \hookrightarrow \end{bmatrix} D \begin{bmatrix} \rightarrow \\ \end{bmatrix} \begin{bmatrix} \rightarrow \\ \end{bmatrix} \begin{bmatrix} \rightarrow \\ \end{bmatrix} \begin{bmatrix} \rightarrow \\ \end{bmatrix}. \tag{4.2}$$

Considering the case where both Krylov spaces only contain finite poles (i.e., 0 in this setting), the LDU factorization of the projected counterpart would have a factorization of the form

$$\begin{bmatrix} \\ \hookrightarrow \end{bmatrix} \begin{bmatrix} \\ \hookrightarrow \end{bmatrix} \begin{bmatrix} \\ \hookrightarrow \end{bmatrix} D \begin{bmatrix} \rightarrow \\ \end{bmatrix} \begin{bmatrix} \rightarrow \\ \end{bmatrix} \begin{bmatrix} \rightarrow \\ \end{bmatrix} \begin{bmatrix} \rightarrow \\ \end{bmatrix}. \tag{4.3}$$

For an existence proof of such a factorization we refer to [38], where it is proved that both the L and U factors in an LDU factorization inherit the lower respectively upper triangular structure of the original matrix. The L and U factors can then be factored as products of essentially 2×2 lower and upper triangular matrices, which are elimination matrices.

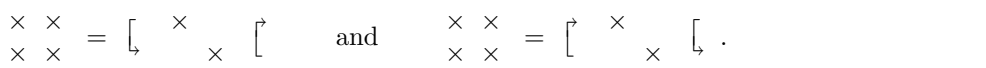
So, in general we need to retrieve factorizations for example of the form



(4.4)

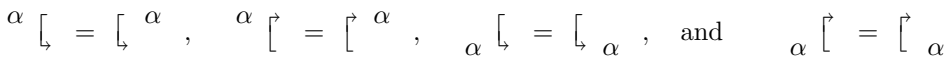
where both the L and U factor are independently built as a zigzag shape of respectively lower and upper eliminators. We will not go into the details, but the ordering of each individual sequence is bond to a selection vector: one selection vector for the lower part linked to $\boldsymbol{\xi}$ and one for the upper part related to $\boldsymbol{\chi}$.

4.3. Manipulating eliminators. We will again manipulate matrices in a factored form. Therefore we need additional operations for dealing with eliminators, just like in Subsection 3.3. The two most important ingredients are the LDU and UDL factorization of a 2×2 matrix, graphically depicted as

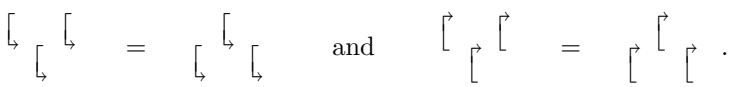


We also need to fuse eliminators:

There are three slightly different settings in which we have to pass scalars through eliminators. In the next equations, the factor α stands in fact for a diagonal matrix equal to the identity, having only one diagonal entry replaced by α. Graphically the passing through reads as



Bare in mind that the eliminators involved in these equalities generally change as well. Eliminators also admit turnover operations just like the rotations[4]



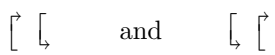
The eliminators have much more flexibility to rearrange their positions compared with rotations. Of course we can change the ordering of eliminators acting on different rows. But we also can do the following swaps, interchanging the position of two eliminators of which one is upper and the other is a lower eliminator. Note that the eliminators are not modified:

[4]We remark that this turnover is almost always possible, see [5], possible breakdowns are inherent to the nature of this problem and cannot be circumvented, see also [26].

Algorithm 2: Extended Tridiagonal Inverse Eigenvalue Problem.

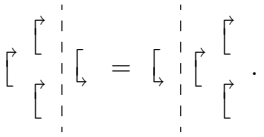
Input: $\mathbf{\Lambda} \in \mathbb{C}^n$, $\mathbf{v}, \mathbf{w} \in \mathbb{C}^n$, ξ
Output: $Z = LDU \in \mathbb{C}^{n\times n}$, V, W *(if required)*
$Z = \text{diag}(\mathbf{\Lambda})$; $D = \text{zeros}(n, n)$; $Q = \text{eye}(n, n)$;
$V = W = \text{eye}(n, n)$ *(if required)*;
Compute G^V that zeroes v_n and compute G^W that zeroes w_n;
Biorthogonalize G^V and G^W;
Apply the similarity transformation on Z, and update its LDU decomposition;
for $k = n - 1 : 2$ **do**
  Compute G^V that zeroes v_k and compute G^W that zeroes w_k;
  Biorthogonalize G^V and G^W;
  Apply the similarity transformation and generate the bulge;
  Chase the bulge according to the four cases in Fig. 4.4,
  thereby update V and W *(if required)*;
end

For a swapping of the following particular configuration



we have to form the product of both matrices and compute the LDU or UDL factorization. We cannot circumvent in this case the introduction of an additional diagonal matrix, which by the passing through can be moved to the outer left or outer right.

These swap operations allow us to *push* lower eliminators *through* a sequence of upper eliminators without altering the rows they act on, the same holds for *pushing* upper eliminators *through* sequences of lower eliminators. For example, it is easily verified that (ignore the possible introduction of an extra diagonal)



It is unnecessary to provide proofs for the operations provided in this section: one can easily show how they are computed by forming the 2×2 resp. 3×3 matrices. We note, however, that some of these operations can potentially be unstable.

4.4. Algorithmic solution. This section elaborates on Algorithm 2 to solve Problem 4.1 and is split in several subsections separating zero, infinite, and mixed pole combinations. Again, instead of operating on the dense matrix directly we represent it in factored form as presented in Section 4.2.

Before initiating our discussion, we would like to recall that in the previous section we always used unitary transformations and thus the Hermitian conjugate and the inverse coincide. Here, however, we recurrently use triangular matrices, either upper or lower. The Hermitian conjugate of such matrices makes upper become lower and lower become upper triangular; the inverse on the contrary does not affect the upper and lower structural constraint. We make an effort to overcome any ambiguity as much as possible.

Whereas in Section 3 we confined ourselves to unitary transformations, this must be loosened here. We will enforce the first columns of W and V to be $\mathbf{w}$ and $\mathbf{v}$ by combining unitary, upper-, and lower triangular operations to carry out dedicated tasks destroying neither the first columns of W and V nor the desired structure of the matrix of recurrences. In general we name any essentially 2×2 matrix embedded in the identity and acting on two successive rows or columns a *core transformation*, which can be a rotator, an eliminator, or a combination.

More precisely, each core transformation to be executed has specified tasks. In each step the first two core transformations executed (left and right) must annihilate elements in the weight vector and must be biorthogonal. The core transformations applied on both sides in the chasing procedure must annihilate bulges and secondly not destroy the biorthogonality. Each core transform encompasses thus two tasks: zero creation and biorthogonality enforcement. To easily achieve this, we will decompose each core transform in a QR or QL decomposition. The unitary factor will impose the zero structure, the upper triangular or lower triangular factors will enforce the biorthogonality; R for the first (left and right) transforms, L in the chasing part.

4.4.1. Only poles at infinity. We initiate our description with the simplest case, namely two classical Krylov spaces. The generic outcome should thus be a nonsymmetric tridiagonal matrix. For this setting it is instructive to first describe the flow of the algorithm by operating directly on the dense matrix. In Section 4.4.2 we translate this to the new factorized representation.

Figure 4.2(a) shows the initial setting. The matrix $Z_0 = \Lambda$ is enclosed by the two weight vectors: $\mathbf{w}_0 = \mathbf{w}$ on its left, and $\overline{\mathbf{v}}_0 = \overline{\mathbf{v}}$ on top. The vector $\overline{\mathbf{v}}$ is intentionally blessed with the conjugate such that the operations applied on the weight vectors can also be applied immediately on the inner matrix Z_0. In the first step, the weights w_5 and $\overline{v}_5$ are annihilated by a left $(G^W_{0,4})^H$ and a right $G^V_{0,4}$ rotation. The subscripts (i, j) in the notation read as follows: i denotes the step, where each step annihilates one element in both weight vectors; and j points to the rows/columns j and $j + 1$ affected by the operation. The meaning of the superscripts is clear from the context. The matrices Z_i denote the outcomes after each step. As a result we get $(G^W_{0,4})^{-1}\,\mathbf{w}_0 = (G^W_{0,4})^H\,\mathbf{w}_0 = \tilde{\mathbf{w}}_0$ and $(G^V_{0,4})^{-1}\,\mathbf{v}_0 = (G^V_{0,4})^H\,\mathbf{v}_0 = \tilde{\mathbf{v}}_0$, both having as final element zero. The result of applying these transformations on Z_0: $(G^W_{0,4})^H\,Z_0\,G^V_{0,4}$ as well is depicted in Figure 4.2(b). To complete the first step it remains to enforce biorthogonality between $G^W_{0,4}$ and $G^V_{0,4}$, this is achieved by assigning $W_0 = G^W_{0,4}(L_{0,4}D_{0,4})^{-H}$ and $V_0 = G^V_{0,4}U^{-1}_{0,4}$, where $(G^W_{0,4})^H\,G^V_{0,4} = L_{0,4}\,D_{0,4}\,U_{0,4}$ is an LDU decomposition. Indeed $W_0^H V_0 = (L_{0,4}D_{0,4})^{-1}\,(G^W_{0,4})^H\,G^V_{0,4}\,U^{-1}_{0,4} = I$ and we get $Z_1 = W_0^H Z_0 V_0 = (L_{0,4}D_{0,4})^{-1}\,(G^W_{0,4})^H\,Z_0\,G^V_{0,4}U^{-1}_{0,4}$. The matrices W_i and V_j gather all transformations executed in a single step. Fortunately also the biorthogonality reinforcement does not destroy the created zeros in the weight vectors as

$$
\begin{aligned}
W_0^{-1}\mathbf{w}_0 &= (L_{0,4}D_{0,4})^H\,(G^W_{0,4})^{-1}\,\mathbf{w}_0 = (L_{0,4}D_{0,4})^H\,\tilde{\mathbf{w}}_0 = \mathbf{w}_1\\
V_0^{-1}\mathbf{v}_0 &= U_{0,4}(G^V_{0,4})^{-1}\,\mathbf{v}_0 = U_{0,4}\tilde{\mathbf{v}}_0 = \mathbf{v}_1,
\end{aligned}
$$

both having the trailing vector element zero since $(L_{0,4}D_{0,4})^H$ and $U_{0,4}$ are upper triangular. Figure 4.2(b) shows the result after the entire first step.

Figure 4.2(c) shows the result after the first tasks in step 2. We see –neglect the core transformations for a moment– the outcome of annihilating elements in the weight vectors as well as the biorthogonality reinforcement. Zeroing an element in $\mathbf{w}_1$ and $\mathbf{v}_1$ is achieved by transformations $G^W_{1,3}$ and $G^V_{1,3}$, we get $(G^W_{1,3})^{-1}\,\mathbf{w}_1 =$

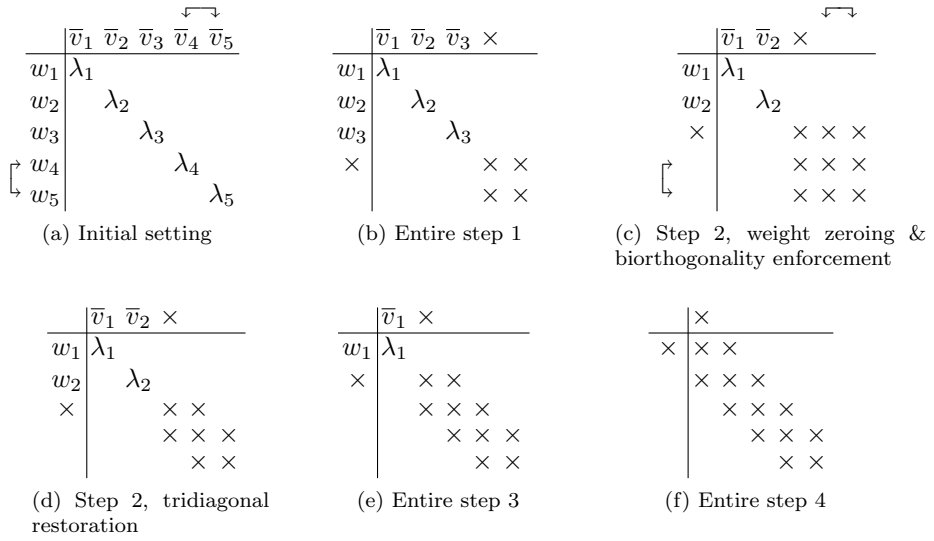
$(G^W_{1,3})^H \mathbf{w}_1 = \tilde{\mathbf{w}}_1$ and $(G^V_{1,3})^{-1} \mathbf{v}_1 = (G^V_{1,3})^H \mathbf{v}_1 = \tilde{\mathbf{v}}_1$, having the penultimate and final element zero. Again, as in step 1, we update the transformation matrices to get them biorthogonal as follows $\tilde{W}_1 = G^W_{1,3}(L_{1,3}D_{1,3})^{-H}$ and $\tilde{V}_1 = G^V_{1,3}U^{-1}_{1,3}$, where $(G^W_{1,3})^H G^V_{1,3} = L_{1,3} D_{1,3} U_{1,3}$ is again an LDU decomposition. Executing $\tilde{W}_1^H Z_1 \tilde{V}_1$ creates, however, unwanted elements in the second sub- and second superdiagonal. Next we need to remove these undesired nonzero elements in the outer corners of the lower right dense 3×3 block to revert back to the tridiagonal structure. This removal is referred to as the chasing procedure. Executing the two rotations depicted in Figure 4.2(c) accomplishes this, name them $G^W_{1,4}$ and $G^V_{1,4}$. We have now $\tilde{Z}_1 = (G^W_{1,4})^H \tilde{W}_1^H Z_1 \tilde{V}_1 G^V_{1,4}$ of tridiagonal form as in Figure 4.2(d). Unfortunately, we cannot apply the previous trick to reenforce the biorthogonality as utilizing the factors of the LDU decomposition would destroy the just reestablished tridiagonal structure. Using, however, the UDL decomposition, with U upper, D diagonal, and L lower triangular and letting $W_1 = \tilde{W}_1 G^W_{1,4} (U_{1,4}D_{1,4})^{-H}$ and $V_1 = \tilde{V}_1 G^V_{1,4} L^{-H}_{1,4}$ where $(\tilde{W}_1 G^W_{1,4})^H (\tilde{V}_1 G^V_{1,4}) = U_{1,4} D_{1,4} L_{1,4}$ we get the biorthogonality as $W_1^H V_1 = (U_{1,4}D_{1,4})^{-1} (\tilde{W}_1 G^W_{1,4})^H (\tilde{V}_1 G^V_{1,4}) L^{-1}_{1,4} = I$. Moreover, by considering $Z_2 = W_1^H Z_1 V_1 = (U_{1,4}D_{1,4})^{-1} (\tilde{W}_1 G^W_{1,4})^H Z_0 (\tilde{V}_1 G^V_{1,4}) L^{-1}_{1,4} = (U_{1,4}D_{1,4})^{-1} \tilde{Z}_1 L^{-1}_{1,4}$, we see that the matrix Z_2 must be of tridiagonal form, as $(U_{1,4}D_{1,4})^{-1}$ is upper triangular, $L^{-1}_{1,4}$ is lower triangular and they only act on the last two rows and columns.

In fact, one has everything now to run the algorithm to the end. Every similarity is executed in the same manner. In step 3, however, the biorthogonality enforcement in the chasing step will introduce new bulges, positioned lower, which in turn need to be removed by another chase step. In general step i requires i-1 chasing steps to remove the bulges and arrive at the tridiagonal form. For completeness, step 3 and step 4 are visualized in Figures 4.2(e) and 4.2(f).

To summarize, each core transformation used in this procedure, is decomposed as either a unitary–upper triangular, or unitary–lower triangular product. The LDU decomposition is used for imposing biorthogonality while retaining the zero structure of the weight vectors. The UDL decomposition is needed for the biorthogonality enforcement in the chasing procedure. This subtle difference between using the LDU and UDL factorization is due to the difference between inverting and taking the conjugate transpose, namely W_i^{-1} and V_i^{-1} need to be applied on the weight vectors, whereas W_i^H and V_i are applied on the matrix Z_i.

4.4.2. Only poles at infinity – factorized representation. Let us now reconsider the previous algorithm, but work directly on the factored representation of the tridiagonal matrix as in (4.2). This will help us to describe the forthcoming algorithm dealing with finite and infinite poles. The main difference between this and the Arnoldi algorithm from Section 3.4 is the appearance of a whole bunch of intermediate swappings of lower and upper triangular eliminators. These swaps are essential to repositioning the eliminators so that all lower and upper eliminators are gathered allowing us to carry out turnovers and fusions.

Again, we will deal with a 5×5 example, initially looking as in Figure 4.3(a). First we compute the rotations $G^W_{0,4}$ and $G^V_{0,4}$ that annihilate w_5 and v_5. To biorthogonalize them we compute the LDU factorization of $(G^W_{0,44})^H G^V_{0,4}$ and update $G^W_{0,4}$ and $G^V_{0,4}$ accordingly. The essentially 2×2 matrices W_0 and V_0 contain all the information to complete the first step. As we wish to store a factored representation as in (4.2), we write W_0 and V_0 as LDU decompositions and get a matrix product like in Figure 4.3(b), where $W_0 = L^W_{0,4} D^W_{0,4} U^W_{0,4}$ and $V_1 = L^V_{0,4} D^V_{0,4} U^V_{0,4}$. We can reorganize the lower and



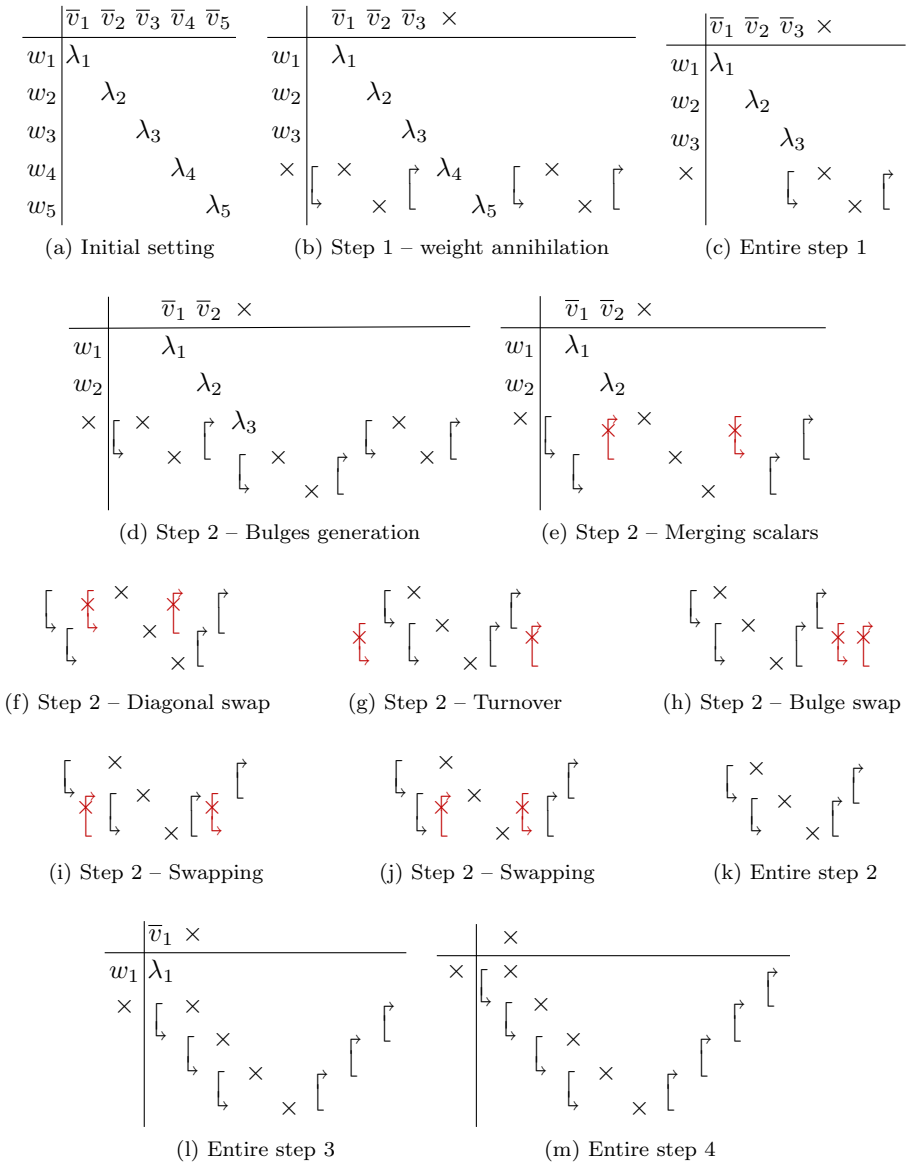
(a) Initial setting (b) Entire step 1 (c) Step 2, weight zeroing & biorthogonality enforcement

(d) Step 2, tridiagonal restoration (e) Entire step 3 (f) Entire step 4

Fig. 4.2: Inverse eigenvalue problem having only poles at infinity (steps 1 to 4).

upper triangular factors by some passing throughs, swappings, and finally some fusions to unite respectively the upper and lower triangular factors leading to Figure 4.3(c). More precisely, let $Z_0 = L_0 D_0 U_0$ denote the LDU factorization of the original matrix Z_0, i.e., both L_0 and U_0 equal the identity and D_0 is diagonal. After annihilating the weights and applying the corresponding transformations on Z_0 we have $W_0^H Z_0 V_0$, or in entirely factored form we have $(L_{0,4}^W D_{0,4}^W U_{0,4}^W)^H L_0 D_0 U_0 (L_{0,4}^V D_{0,4}^V U_{0,4}^V)$. Regrouping the individual factors gives us $(U_{0,4}^W)^H \left((D_{0,4}^W)^H (L_{0,4}^W)^H L_0 D_0 U_0 L_{0,4}^V D_{0,4}^V\right) U_{0,4}^V$. Passing the diagonals $D_{0,4}^W$ and $D_{0,4}^V$ through to join the middle diagonal D_0 one can get rid of them. Next one swaps the upper and lower triangular factors so that all lower triangular factors appear left and the upper triangular factors appear on the right: $(U_{0,4}^W)^H \tilde{L}_0 \tilde{D}_0 \tilde{U}_0 U_{0,4}^V$. After two more fusions we get $Z_1 = L_1 D_1 U_1$ in factored form. One can directly compute the LDU factorization of $W_0^H Z_0 V_0$. However, for the sake of a unified explanation for all steps we choose to present this more intricate way here. The implementation of the algorithm uses of course the cheaper way.

In the second step we begin again with the computation of the annihilating rotators and their biorthogonalization. Factoring these essentially 2×2 matrices leads to Figure 4.3(d). The diagonal matrices appearing in the LDU factorizations are immediately passed through the eliminators and are merged with the main central diagonal. We explicitly assume from now on that this is done every time an LDU or UDL factorization is computed, e.g., also in the case where two eliminators (an upper and lower triangular one) acting on the same rows are swapped. As a result we will not depict these diagonals anymore. Little reordering leads to Figure 4.3(e). There are now two undesired eliminators, which will consistently be referred to as the *bulges* in the remainder of the text. These transformations are marked by a ×. We have now two lower eliminators $L_{1,3}$, $L_{1,4}$, and the bulge E_l on the left and two upper eliminators $U_{1,4}$, $U_{1,3}$, and another bulge on the right E_r. We will chase the bulges down to merge them with $L_{1,3}$ or $U_{1,4}$.



(a) Initial setting (b) Step 1 – weight annihilation (c) Entire step 1

(d) Step 2 – Bulges generation (e) Step 2 – Merging scalars

(f) Step 2 – Diagonal swap (g) Step 2 – Turnover (h) Step 2 – Bulge swap

(i) Step 2 – Swapping (j) Step 2 – Swapping (k) Entire step 2

(l) Entire step 3 (m) Entire step 4

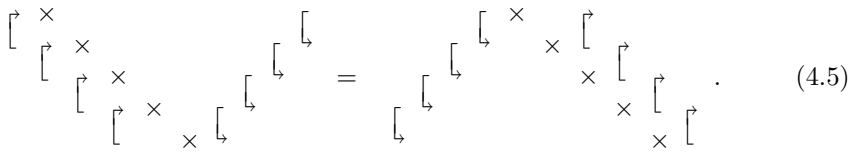
Fig. 4.3: Inverse eigenvalue problem having only poles at infinity – factorized.

The core ideas of this chasing procedure are identical to those used in the in the Arnoldi setting: bring the bulges down by a turnover, next bring them to the other side by similarities, and finally fuse them with the bottom transformations. To carry this out in this configuration we need some additional swaps from upper-lower to lower-upper factorizations. First the bulges swap their positions, we arrive at Figure 4.3(f): the essential piece of information is found when comparing the arrow-tips of the previous and this figure. In the next figures the weights are removed and only the effective part of the matrix and its factorization are presented. Next all transfor-

mations are in the correct position to make two turnovers to move the bulges down one row, see Figure 4.3(g). In the former case, the Arnoldi setting, a similarity was executed, after which we could remove the disturbing rotation by a fusion. Here we follow a similar procedure, first we bring one bulge to the other side by a similarity and swap the positions of the two bulges[5], this is shown in Figure 4.3(h). Finally, we bring the outer bulge back to the other side by a similarity. We remark that both executed similarity transformations do not affect the already generated zeros in the vectors of weights. Now we are ready to dispose of the bulges. Push both bulges through the sequences of lower and upper eliminators; this involves two swaps as shown in Figures 4.3(i) and (j). Now the bulges are in the same position as at the begin of the chasing, but one row down. To complete step 2 we swap the bulges again and merge them with the lower and upper triangular factors of the representation, see Figure 4.3(k).

In Figures 4.3(l) and (m) the outcome of the entire third and fourth step are visualized. We end with an LDU factorization of our sought tridiagonal matrix.

4.4.3. Only poles at zero. Before tackling the general case (4.4), let's discuss (4.3). If we have only finite poles with $\xi_i = 0$ and $\chi_i = 0$ we can use an algorithm analogue to the one described in Subsection 4.4.2. We only have to replace the LDU factorization by an UDL factorization and every upper eliminator by a lower eliminator and vice versa. Only to retrieve the bulges by annihiliting weights and restoring the biorthogonality by factoring $G_i^W (G_i^V)^H$ we still need to employ the LDU factorization in order to retain the sparsity of the weight vectors. At the end we would retrieve a matrix factored as the left-hand side in (4.5), but this can easily be transformed by some swaps to the desired format on the right-hand side of (4.5)



(4.5)

If, however, we directly want to have the second representation, then we start the same way as in Section 4.4.2 until we reach the state of Figure 4.3(f). The bulge rotators, the ones perturbing the pattern are now not positioned on the inside but on the outside. These bulges will move to the inner side of the factorization after the turnover. We swap them through the diagonal and push them through the ascending respectively descending sequence of eliminators. They are now again positioned on the outside. Finally we bring one to the other side (by similarity), swap the bulges again and bring one bulge back to end up in the same situation as at the beginning of the chasing but one row down.

4.4.4. General case. We now combine poles at zero and infinity to the general case. From Subsection 3.4.3 we know that the crucial steps for the generalization to a mixture of finite and infinite poles are the change from one to the other at the final chasing step and the fact that bends in the zigzag shapes move up one row every time we chase a bulge through them. Besides this the essentials remain the same: in each step we carry out a turnover to move the bulges down and bring them over to the

[5] Theoretically it doesn't matter which bulge is brought to the other side first. Our implementation tries both variants: bringing the left bulge to the right as well as the right bulge to the left. The variant delivering a better biorthogonality of W and V is taken.

other side by similarities. We require, however, a bunch of intermediate swappings to reposition the upper and lower eliminators in order to gather them. We refrain from detailing the operations mathematically, but will only depict the essence (not showing the weights) of the vital steps graphically.

The bulges are generated identically as before: compute weight annihilating rotators and biorthogonalize them. Next, we compute the LDU factorization of the two 2×2 matrices, swap the inner eliminators and merge all the diagonals to arrive at Figure 4.4(a). We now have two new eliminators on each side. Four eliminators act on the top rows amongst these: two, one on each side, are the bulges. The remaining eliminators make up the part of the zigzag shape forming L and U. More precisely, first consider the L-factor. If the first entry in $\boldsymbol{\xi}$ is ∞, then the inner eliminator (positioned on the right side of the L-factor) is the bulge and the outer one is part of L. If the first entry in $\boldsymbol{\xi}$ is 0, then the outer eliminator is the bulge. The same holds for the matrix U on the right-hand side. The bulge is the inner eliminator if χ_1 equals ∞ and the outer one if $\chi_1 = 0$, the remaining eliminators form the zigzag sequence in the factorization of U.

Chasing the bulges starts, as always, with moving them down one row. If there is no bend, we are in one of the two situations depicted in Figure 4.4(b), here the arrows are omitted since the figure holds for both the upper (U-factor) and lower sequence (L-factor) of eliminators. The bulges are marked with a $\times$. The bulges change side, according to the changes in the zigzag sequence. An easy rule of thumb says that only the forthcoming two eliminators (positioned one and two rows lower) determine which eliminator becomes the bulge: if the next two are ordered descendingly, the bulge is the right one, if they are ordered ascendingly the bulge is found on the left. It is interesting to realize that this reasoning is not feasible anymore when reaching the bottom of a pattern, simply because there are no two eliminators left. In this case the choice of the bulge is unrestricted. Figure 4.4(c) applies again to both types of eliminators (lower and upper) and shows which eliminator becomes the bulge in case of a turn. This behavior is quite similar to Figure 3.8.

The remaining task is to bring the bulges to the other side of the zigzag sequence, which is accomplished by similarities and swappings. The following four situations might arise.

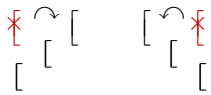
(A) **Both bulges outside.** This concurs with the infinite poles case. The bulges follow the arrows as in Figure 4.4(d). First we bring one bulge to the other side (similarity), swap the bulges, and bring the outer one back (similarity). Then we push the bulges through the zigzag shape. Finally we swap the bulges through the diagonal.

(B) **Both bulges inside.** This setting agrees with the finite poles case. The bulges follow the arrows as in Figure 4.4(e). First we swap the bulges through the diagonal simultaneously. We then push both independently through the zigzag sequences on their sides. They appear now on the outside. We bring one to the other side, swap the bulges, and bring the outer one back.

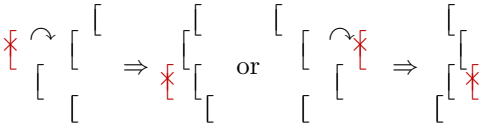
(C) **Left bulge inside, right bulge outside.** Both bulges move to the right and do not have to pass each other, see Figure 4.4(f). We pass the left bulge through the diagonal and the right zigzag sequence. We bring the right bulge to the other side, pass it through the left zigzag sequence and the diagonal. Finally we bring the formerly left bulge back to the left side.

(D) **Left bulge outside, right bulge inside.** Both bulges move to the left and do not have to pass each other, see Figure 4.4(g). We pass the right bulge

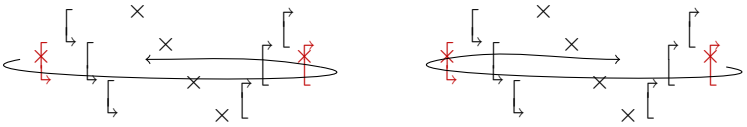
(a) Bulge generation



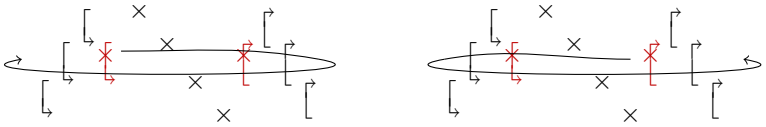
(b) Turnovers – no bend



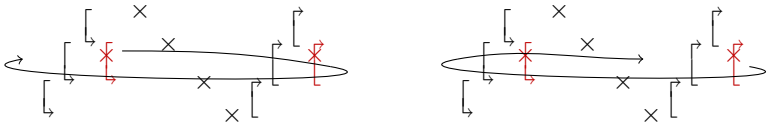
(c) Turnovers – with bend



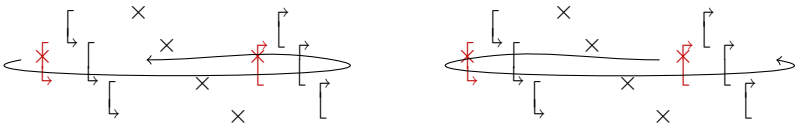
(d) Bulges to the other side (A)



(e) Bulges to the other side (B)



(f) Bulges to the other side (C)



(g) Bulges to the other side (D)

(h) Merging the bulges

Fig. 4.4: General case.

through the diagonal and the left zigzag sequence. We bring the left bulge to the other side, pass it through the right zigzag sequence and the diagonal. Finally we bring the formerly right bulge back to the right side.

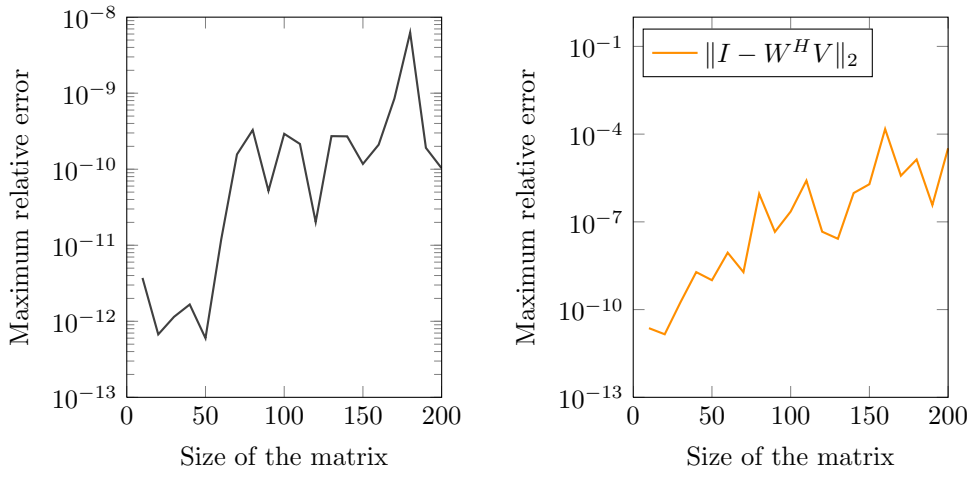
If one analyzes Figure 4.4(d)-(g), then one observes that we have actually two cases

for the left bulge and two cases for the right bulge. However, it is from an implementational point of view advantageous to divide the possible cases in the four described above. For instance, the cases (A) and (B) contain the swapping of the two bulges, whereas in the cases (C) and (D) such a swapping is not necessary.

We have now completed one chasing step. After sufficient steps we have reached the bottom of both zigzag shapes, and we need to merge a bulge with the L factor, and another bulge with the U factor. We are now in a situation like in Figure 4.4(h). We have now the flexibility to form new bends at the end of our zigzag sequences, depending on ξ and χ. On the left (L-factor), one of the bottom eliminators needs to be taken as bulge and on the right (U-factor) also. For a last time in this chasing procedure we bring the bulges to the other sides by similarities and swaps if necessary, and finally merge some eliminators left and right.

4.5. Numerical experiments. Two tests were executed. First the accuracy and speed were examined, when the eigenvectors are desired. Second, both algorithms proposed in this paper were compared with each other.

4.5.1. Speed and accuracy. We have implemented the algorithm as described in the last subsection. As in Section 3.6 we tested our algorithm with random eigenvalues in the complex plane, random complex weight vectors, and random zigzag patterns. We did again five tests per dimension. The error on the eigenvalues is computed as in Section 3.6.



(a) Relative accuracy of the eigenvalues.

(b) Accuracy of the biorthogonality

Fig. 4.5: Relative accuracy.

We observe that the accuracy of this more general algorithm is not as good as for the classical Arnoldi setting. While the accuracy of the eigenvalues is still acceptable, see Figure 4.5(a), the accuracy of the weights is not, see Figure 4.5(b). The biorthogonality is also not good for dimensions larger than 100. As the algorithm from the last section, this algorithm is also of cubic complexity, as Figure 4.6 shows. The $\mathcal{O}(n^2)$ updates of V and W change each $\mathcal{O}(n)$ entries and thus the algorithm is of cubic complexity. Omitting the computation of V and W reduces the complexity to quadratic.

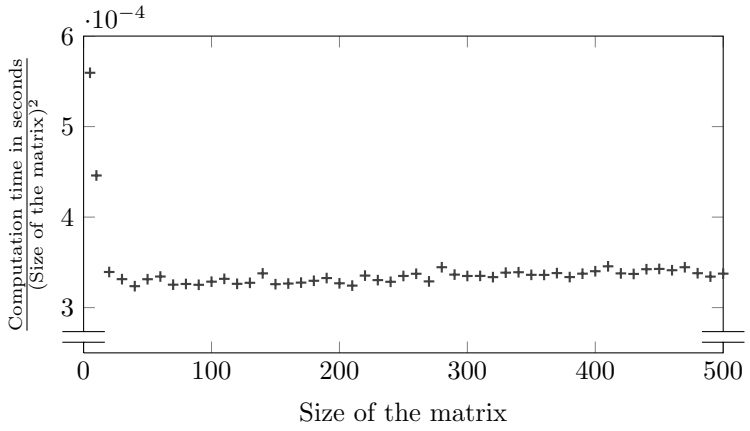


Fig. 4.6: Computational complexity.

4.5.2. Comparison of both methods. Both algorithms are compared when only the matrix of recurrences is desired. In this case the first algorithm is cubic, whereas the second algorithm is of quadratic complexity. Table 4.1 depicts the problem sizes, ranging from 10 to 500, the maximum error on the eigenvalues, the time needed to retrieve the matrix of recurrences, and the time divided by either the problem size squared or tripled to illustrate that both implementations are of the correct complexity. The table has two successive rows dedicated to the same problem size but alternatingly assigned to first and the second algorithm.

	Size n	Max. Eigv. Error	Time	Time/n^2	Time/n^3
Alg. 1	10	7.9 e−16	2.4 e−02		2.4 e−05
Alg. 2		5.9 e−14	5.3 e−02	5.3 e−04	
Alg. 1	20	4.0 e−15	9.9 e−02		1.2 e−05
Alg. 2		2.9 e−13	1.4 e−01	3.6 e−04	
Alg. 1	50	1.0 e−16	8.3 e−01		6.7 e−06
Alg. 2		1.8 e−12	7.9 e−01	3.1 e−04	
Alg. 1	100	2.8 e−15	5.3 e+00		5.3 e−06
Alg. 2		5.4 e−12	3.3 e+00	3.3 e−04	
Alg. 1	150	5.0 e−15	1.7 e+01		4.9 e−06
Alg. 2		3.8 e−11	7.3 e+00	3.2 e−04	
Alg. 1	200	6.7 e−15	4.4 e+01		5.4 e−06
Alg. 2		3.8 e−10	1.4 e+01	3.5 e−04	
Alg. 1	250	1.3 e−14	8.1 e+01		5.2 e−06
Alg. 2		5.0 e−10	2.1 e+01	3.3 e−04	
Alg. 1	300	3.4 e−15	1.4 e+02		5.0 e−06
Alg. 2		3.1 e−10	3.0 e+01	3.3 e−04	
Alg. 1	350	1.8 e−14	2.1 e+02		4.9 e−06
Alg. 2		1.0 e−08	4.1 e+01	3.4 e−04	
Alg. 1	400	2.2 e−14	3.6 e+02		5.7 e−06
Alg. 2		1.5 e−08	5.4 e+01	3.4 e−04	
Alg. 1	450	1.1 e−13	5.3 e+02		5.8 e−06
Alg. 2		6.5 e−09	6.8 e+01	3.4 e−04	
Alg. 1	500	2.4 e−14	6.8 e+02		5.4 e−06
Alg. 2		1.8 e−05	8.5 e+01	3.4 e−04	

Table 4.1: Comparing both algorithms w.r.t. accuracy and computational complexity.

The results clearly indicate that the non-unitary algorithm is much faster than the unitary variant, but on the other hand there is clearly a price to be paid: the first algorithm is more accurate than the second one.

5. Conclusions. In this paper we presented two algorithms for solving two inverse eigenvalue problems. Given eigenvalues, poles, and weights algorithms were presented to reconstruct particular matrices of specific structure. This structure links to rational Arnoldi and orthogonal rational functions, and to rational nonsymmetric Lanczos and biorthogonal rational functions.

Both algorithms operate on factored forms of the resulting matrices and rely on executing similarity transformations with 2×2 matrices. The first algorithm is slower, but more accurate as it executes unitary similarities, the second algorithm is faster, because of a more compact representation of the matrix, but is less accurate due to the loss of unitarity.

Future research involves generalizing the second algorithm to deal with nonzero finite poles, and also attention will be paid to breakdowns and nearly breakdowns. A possible solution is to use look-ahead techniques. This, however, has a significant impact on the algorithm and requires solid heuristics to foresee numerical problems.

REFERENCES

[1] G. S. Ammar and W. B. Gragg, *$\mathcal{O}(n^2)$ reduction algorithms for the construction of a band matrix from spectral data*, SIAM Journal on Matrix Analysis and Applications, 12 (1991), pp. 426–432.

[2] G. S. Ammar, W. B. Gragg, and L. Reichel, *Constructing a unitary Hessenberg matrix from spectral data*, in Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, G. H. Golub and P. Van Dooren, eds., vol. 70 of Computer and Systems Sciences, Springer-Verlag, Berlin, Germany, 1991, pp. 385–395.

[3] ———, *Downdating of Szegő polynomials and data-fitting applications*, Linear Algebra and its Applications, 172 (1992), pp. 315–336.

[4] G. S. Ammar and C. He, *On an inverse eigenvalue problem for unitary Hessenberg matrices*, Linear Algebra and its Applications, 218 (1995), pp. 263–271.

[5] J. Aurentz, R. Vandebril, and D. S. Watkins, *Fast computation of the zeros of a polynomial via factorization of the companion matrix*, SIAM Journal on Scientific Computing, 35 (2013), pp. A255–A269.

[6] D. L. Boley and G. H. Golub, *A survey of matrix inverse eigenvalue problems*, Inverse Problems, 3 (1987), pp. 595–622.

[7] A. Bultheel and B. L. R. De Moor, *Rational approximation in linear systems and control*, Journal of Computational and Applied Mathematics, 121 (2000), pp. 355–378.

[8] A. Bultheel, P. González-Vera, E. Hendriksen, and O. Njåstad, *Orthogonal Rational Functions*, vol. 5 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, United Kingdom, 1999.

[9] A. Bultheel and M. Van Barel, *Vector orthogonal polynomials and least squares approximation*, SIAM Journal on Matrix Analysis and Applications, 16 (1995), pp. 863–885.

[10] A. Bunse-Gerstner and L. Elsner, *Schur parameter pencils for the solution of the unitary eigenproblem*, Linear Algebra and its Applications, 154-156 (1991), pp. 741–778.

[11] M. T. Chu and G. H. Golub, *Inverse Eigenvalue Problems: Theory, Algorithms and Applications*, Numerical Mathematics & Scientific Computations, Oxford University Press, New York, USA, 2005.

[12] S. Elhay, G. H. Golub, and J. Kautsky, *Updating and downdating of orthogonal polynomials with data fitting applications*, SIAM Journal on Matrix Analysis and Applications, 12 (1991), pp. 327–353.

[13] V. Faber and T. Manteuffel, *Necessary and sufficient conditions for the existence of a Conjugate Gradient method*, SIAM Journal on Numerical Analysis, 21 (1984), pp. 352–362.

[14] H. Fassbender, *On numerical methods for discrete least-squares approximation by trigonometric polynomials*, Mathematics of Computation, 66 (1997), pp. 719–741.

[15] G. H. Golub and G. Meurant, *Matrices, Moments and Quadrature with Applications*, Princeton Series in Applied Mathematics, Princeton University Press, 2010.

[16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, USA, third ed., 1996.

[17] W. B. Gragg and W. J. Harrod, *The numerically stable reconstruction of Jacobi matrices from spectral data*, Numerische Mathematik, 44 (1984), pp. 317–335.

[18] A. B. J. Kuijlaars, *Which eigenvalues are found by the Lanczos method?*, SIAM Journal on Matrix Analysis and Applications, 22 (2000), pp. 306–321.

[19] J. Liesen and Z. Strakoš, *On optimal short recurrences for generating orthogonal Krylov subspace bases*, SIAM Review, 50 (2008), pp. 485–503.

[20] T. Mach, M. S. Pranić, and R. Vandebril, *Computing approximate extended Krylov subspaces without explicit inversion*. Accepted for publication in Electronic Transactions on Numerical Analysis.

[21] C. Mertens and R. Vandebril, *Multiple recursions and the associated matrix structures stemming from normal matrices*. submitted for publication, 2012.

[22] B. Ninness and F. Gustafsson, *A unifying construction of orthonormal bases for system identification*, IEEE Transactions on Automatic Control, 42 (1997), pp. 515–521.

[23] L. Reichel, *Fast QR-decomposition of Vandermonde-like matrices and polynomial least squares approximation*, SIAM Journal on Matrix Analysis and Applications, 12 (1991), pp. 552–564.

[24] L. Reichel, G. S. Ammar, and W. B. Gragg, *Discrete least squares approximation by trigonometric polynomials*, Mathematics of Computation, 57 (1991), pp. 273–289.

[25] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, Manchester, United Kingdom, 1992.

[26] ———, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, Pennsylvania, USA,

second ed., 2003.
[27] M. Van Barel and A. Bultheel, *A new approach to the rational interpolation problem: the vector case.*, Journal of Computational and Applied Mathematics, 33 (1990), pp. 331–346.
[28] ———, *A parallel algorithm for discrete least squares rational approximation*, Numerische Mathematik, 63 (1992), pp. 99–121.
[29] ———, *Discrete linearized least squares approximation on the unit circle*, Journal of Computational and Applied Mathematics, 50 (1994), pp. 545–563.
[30] ———, *Orthonormal polynomial vectors and least squares approximation for a discrete inner product*, Electronic Transactions on Numerical Analysis, 3 (1995), pp. 1–23.
[31] ———, *Updating and downdating of orthonormal polynomial vectors and some applications*, in Structured Matrices in Mathematics, Computer Science, and Engineering II, V. Olshevsky, ed., vol. 281 of Contemporary Mathematics, American Mathematical Society, Providence, Rhode Island, USA, 2001, pp. 145–162.
[32] M. Van Barel and A. A. Chesnokov, *A method to compute recurrence relation coefficients for bivariate orthogonal polynomials by unitary matrix transformations*, Numerical Algorithms, 55 (2010), pp. 383–402.
[33] M. Van Barel, D. Fasino, L. Gemignani, and N. Mastronardi, *Orthogonal rational functions and diagonal plus semiseparable matrices*, in Advanced Signal Processing Algorithms, Architectures, and Implementations XII, F. T. Luk, ed., vol. 4791 of Proceedings of SPIE, Bellingham, Washington, USA, 2002, pp. 167–170.
[34] ———, *Orthogonal rational functions and structured matrices*, SIAM Journal on Matrix Analysis and Applications, 26 (2005), pp. 810–829.
[35] H. A. van der Vorst, *BiCGSTAB: A fast and smoothly converging variant of BiCG for the solution of nonsymmetric linear systems*, SIAM Journal on Statistical Computing, 13 (1992), pp. 631–644.
[36] H. A. van der Vorst, *Iterative Krylov methods for large linear systems*, vol. 13 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2003.
[37] R. Vandebril, *Chasing bulges or rotations? A metamorphosis of the QR-algorithm*, SIAM Journal on Matrix Analysis and Applications, 32 (2011), pp. 217–247.
[38] R. Vandebril, M. Van Barel, and N. Mastronardi, *Matrix Computations and Semiseparable Matrices, Volume I: Linear Systems*, Johns Hopkins University Press, Baltimore, Maryland, USA, 2008.
[39] R. Vandebril and D. S. Watkins, *A generalization of the multishift QR algorithm*, SIAM Journal on Matrix Analysis and Applications, 33 (2012), pp. 759–779.
[40] D. S. Watkins, *Some perspectives on the eigenvalue problem*, SIAM Review, 35 (1993), pp. 430–471.
[41] ———, *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*, SIAM, Philadelphia, USA, 2007.